用Python 做文本挖掘

[用 Python 做文本挖掘的流程 1](#_Toc30708913)

[如何用Python网络爬虫爬网页 3](#_Toc30708914)

[https://zhuanlan.zhihu.com/p/34206711 3](#_Toc30708915)

[Python3.6有关beautifulsoup成功安装的经验 31](#_Toc30708916)

[Python3 爬虫配置BeautifulSoup4+lxml+html5lib 31](#_Toc30708917)

[详解Python 用 requests + Beautiful Soup 爬取房天下 35](#_Toc30708918)

[手把手教你如何用 Python 做情感分析 42](#_Toc30708919)

[Espresso-使用TensorFlow进行中文自然语言处理的情感分析（笔记） 54](#_Toc30708920)

[综述论文：情感分析中的深度学习（内容抽象） 61](#_Toc30708921)

# 用 Python 做文本挖掘的流程

[**肖智博**](https://www.zhihu.com/people/xiaozhibo)新加坡南洋理工大学 电气与电子工程博士后

<https://zhuanlan.zhihu.com/p/19630762>

## **收集数据**

* + 数据集。如果是已经被人做成数据集了，这就省去了很多麻烦事
  + 抓取。这个是 Python 做得最好的事情，优秀的包有很多，比如 [scrapy](https://link.zhihu.com/?target=http%3A//scrapy.org)，[beautifulsoup](https://link.zhihu.com/?target=http%3A//www.crummy.com/software/BeautifulSoup/) 等等。

## **预处理**

在预处理部分仅仅针对英文的情况，由于中英文在分词上是不同的机制，所以在处理中文的时候需要根据情况进行，个人经验是在去停词之前分词。这部分有待完善。

中文分词。[jieba](https://link.zhihu.com/?target=https%3A//github.com/fxsjy/jieba) 或者 [Stanford (Chinese) Word Segmenter](https://link.zhihu.com/?target=http%3A//nlp.stanford.edu/software/segmenter.shtml)。jieba 是纯 Python 写的，Stanford 的可以通过 [nltk](https://link.zhihu.com/?target=http%3A//nltk.org/) 调用，[复旦 NLP](https://link.zhihu.com/?target=http%3A//code.google.com/p/fudannlp/) 也可以用 Python 调用。

（对[这里](https://link.zhihu.com/?target=https%3A//lists.cs.princeton.edu/pipermail/topic-models/2013-November/002593.html)的高质量讨论结果的修改，下面的顺序仅限**英文**）

1. 去掉抓来的数据中不需要的部分，比如 HTML TAG，只保留文本。结合 beautifulsoup 和正则表达式就可以了。[pattern.web](https://link.zhihu.com/?target=http%3A//www.clips.ua.ac.be/pages/pattern-web) 也有相关功能。
2. 处理编码问题。没错，即使是英文也需要处理编码问题！由于 Python2 的历史原因，不得不在编程的时候自己处理。英文也存在 unicode 和 utf-8 转换的问题，中文以及其他语言就更不用提了。[这里](https://link.zhihu.com/?target=http%3A//www2.imm.dtu.dk/pubdb/views/edoc_download.php/6095/pdf)有一个讨论，可以参考，当然网上也有很多方案，找到一个适用于自己的最好。
3. 将文档分割成句子。
4. 将句子分割成词。专业的叫法是 tokenize。
5. 拼写错误纠正。[pyenchant](https://link.zhihu.com/?target=http%3A//pythonhosted.org/pyenchant/) 可以帮你！（中文就没有这么些破事！）
6. POS Tagging。nltk 是不二选择，还可以使用 [pattern](https://link.zhihu.com/?target=http%3A//www.clips.ua.ac.be/pages/pattern)。
7. 去掉标点符号。使用正则表达式就可以。
8. 去掉长度过小的单词。len<3 的是通常选择。
9. 去掉 non-alpha 词。同样，可以用正则表达式完成 \W 就可以。
10. 转换成小写。
11. 去掉停用词。Matthew L. Jockers 提供了一份比机器学习和自然语言处理中常用的停词表[更长的停词表](https://link.zhihu.com/?target=http%3A//www.matthewjockers.net/wp-content/uploads/2013/04/uwm-workshop.zip)。[中文的停词表](https://link.zhihu.com/?target=http%3A//www.datatang.com/data/43894) 可以参考这个。
12. lemmatization/stemming。nltk 里面提供了好多种方式，推荐用 wordnet 的方式，这样不会出现把词过分精简，导致词丢掉原型的结果，如果实在不行，也用 snowball 吧，别用 porter，porter 的结果我个人太难接受了，弄出结果之后都根本不知道是啥词了。[MBSP](https://link.zhihu.com/?target=http%3A//www.clips.ua.ac.be/pages/MBSP)也有相关功能。
13. 重新去掉长度过小的词。是的，再来一遍。
14. 重新去停词。上面这两部完全是为了更干净。
15. 到这里拿到的基本上是非常干净的文本了。如果还有进一步需求，还可以根据 POS 的结果继续选择某一种或者几种词性的词。

* [Bag-of-Words](https://link.zhihu.com/?target=http%3A//weibo.com/1657470871/Aicj2oP8G)! nltk 和 [scikit.learn](https://link.zhihu.com/?target=http%3A//scikit-learn.org/stable/) 里面都有很完整的方案，自己选择合适的就好。这里[如果不喜欢没有次序的 unigram 模型](https://link.zhihu.com/?target=http%3A//scikit-learn.org/stable/modules/feature_extraction.html%23limitations-of-the-bag-of-words-representation)，可以自行选择 bi-gram 和 tri-gram 以及更高的 n-gram 模型。nltk 和 sklearn里面都有相关的处理方法。
* 更高级的特征。
  + TF-IDF。这个 nltk 和 sklearn 里面也都有。
  + [Hashing](https://link.zhihu.com/?target=http%3A//scikit-learn.org/stable/modules/feature_extraction.html%23vectorizing-a-large-text-corpus-with-the-hashing-trick)！

## **训练模型**

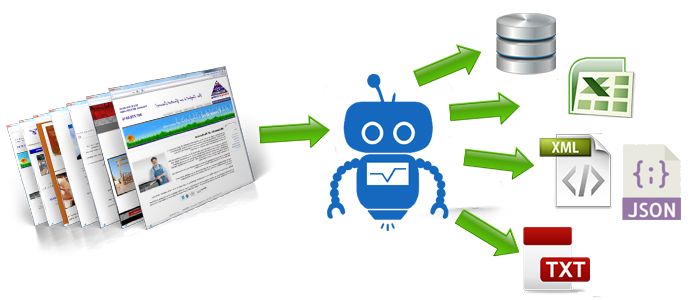
* + 到这里，就根据自己的应用选择合适的学习器就好了。
  + 分类，情感分析等。[sklearn](https://link.zhihu.com/?target=http%3A//scikit-learn.org/stable/supervised_learning.html) 里面很多方法，pattern 里有[情感分析的模块](https://link.zhihu.com/?target=http%3A//www.clips.ua.ac.be/pages/pattern-en%23sentiment)，nltk 中也有一些分类器。
  + 主题发现
    - [NMF](https://link.zhihu.com/?target=http%3A//scikit-learn.org/stable/auto_examples/applications/topics_extraction_with_nmf.html%23example-applications-topics-extraction-with-nmf-py)
    - [(Online) Latent Dirichlet Allocation](https://link.zhihu.com/?target=http%3A//radimrehurek.com/gensim/wiki.html%23latent-dirichlet-allocation)
    - [word2vec](https://link.zhihu.com/?target=http%3A//radimrehurek.com/gensim/models/word2vec.html)
  + 自动文摘。这个自己写吧，没发现什么成型的工具。
* Draw results
  + Matplotlib
  + Tag cloud
  + [Graph](https://link.zhihu.com/?target=http%3A//www.clips.ua.ac.be/pages/pattern-graph)

# 如何用Python网络爬虫爬网页

# <https://zhuanlan.zhihu.com/p/34206711>

[**王树义**](https://www.zhihu.com/people/wang-shu-yi-11)南开大学 情报学博士

你期待已久的Python网络数据爬虫教程来了。本文为你演示如何从网页里找到感兴趣的链接和说明文字，抓取并存储到Excel。



目前主流而合法的网络数据收集方法，主要分为3类：

* 开放数据集下载；
* API读取；
* 爬虫。

前两种方法，我都已经做过一些介绍，这次说说爬虫。

## **网络爬虫概念:**

许多读者对爬虫的定义，有些混淆。咱们有必要辨析一下。维基百科是这么说的：

网络爬虫（英语：web crawler），也叫网络蜘蛛（spider），是一种用来自动浏览[万维网](https://link.zhihu.com/?target=https%3A//zh.wikipedia.org/wiki/%25E4%25B8%2587%25E7%25BB%25B4%25E7%25BD%2591)的[网络机器人](https://link.zhihu.com/?target=https%3A//zh.wikipedia.org/w/index.php%3Ftitle%3D%25E7%25BD%2591%25E7%25BB%259C%25E6%259C%25BA%25E5%2599%25A8%25E4%25BA%25BA%26action%3Dedit%26redlink%3D1)。其目的一般为编纂[网络索引](https://link.zhihu.com/?target=https%3A//zh.wikipedia.org/w/index.php%3Ftitle%3D%25E7%25BD%2591%25E7%25BB%259C%25E7%25B4%25A2%25E5%25BC%2595%26action%3Dedit%26redlink%3D1)。

这问题就来了，你又不打算做搜索引擎，为什么对网络爬虫那么热心呢？

其实，许多人口中所说的爬虫（web crawler），跟另外一种功能“网页抓取”（web scraping）搞混了。

维基百科上，对于后者这样解释：

Web scraping, web harvesting, or web data extraction is data scraping used for extracting data from websites. Web scraping software may access the World Wide Web directly using the Hypertext Transfer Protocol, or through a web browser.

看到没有，即便你用浏览器手动拷贝数据下来，也叫做网页抓取（web scraping）。是不是立刻觉得自己强大了很多？

但是，这定义还没完：

While web scraping can be done manually by a software user, the term typically refers to automate processes implemented using a bot or web crawler.

也就是说，用爬虫（或者机器人）**自动**替你完成网页抓取工作，才是你真正想要的。

数据抓下来干什么呢？一般是先存储起来，放到数据库或者电子表格中，以备检索或者进一步分析使用。

所以，你真正想要的功能是这样的：找到链接，获得Web页面，抓取指定信息，存储。这个过程有可能会往复循环，甚至是滚雪球。你希望用自动化的方式来完成它。

了解了这一点，你就不要老盯着爬虫不放了。爬虫研制出来，其实是为了给搜索引擎编制索引数据库使用的。你为了抓取点儿数据拿来使用，已经是大炮轰蚊子了。

要真正掌握爬虫，你需要具备不少基础知识。例如HTML, CSS, Javascript, 数据结构……有了这个基础，你就能比之前更轻松获取数据了。特别是对“文科生”的很多应用场景来说，非常有用。这就是**赋能**。而且，再进一步深入理解爬虫的工作原理，也变得轻松许多。

Python语言的重要特色之一，就是可以利用强大的软件工具包（许多都是第三方提供）。你只需要编写简单的程序，就能自动解析网页，抓取数据。

本文给你演示这一过程。

## **目标**

要抓取网页数据，我们先制订一个小目标。

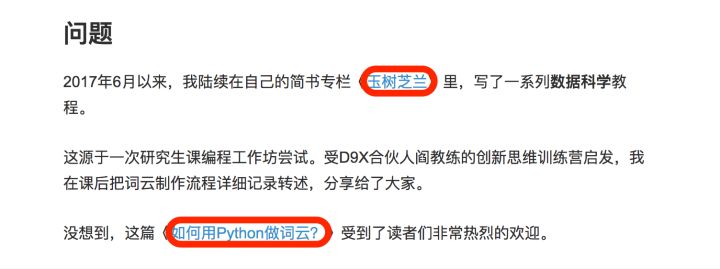
目标不能太复杂。但是完成它，应该对你理解抓取（Web Scraping）有帮助。

就选择我最近发布的一篇简书文章作为抓取对象好了。题目叫做《[如何用《玉树芝兰》入门数据科学？](https://link.zhihu.com/?target=https%3A//www.jianshu.com/p/85f4624485b9)》。

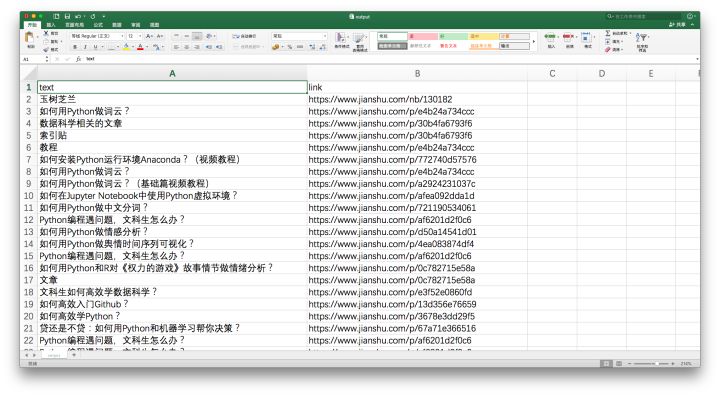


这篇文章里，我把之前的发布的数据科学系列文章做了重新组织和串讲。

文中包含很多之前教程的标题和对应链接。例如下图红色边框圈起来的部分。



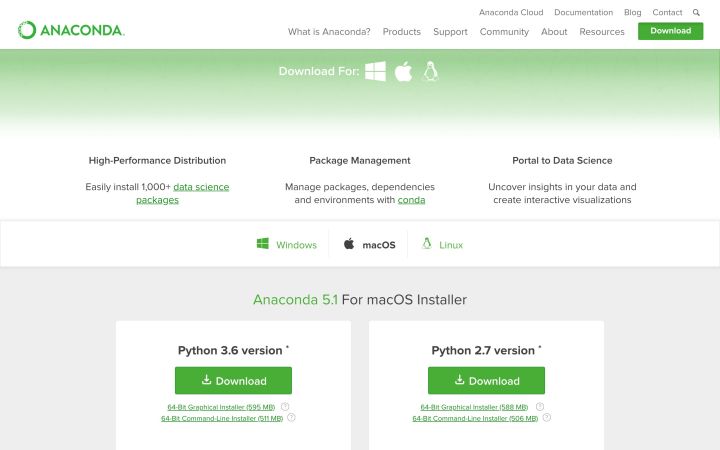
假设你对文中提到教程都很感兴趣，希望获得这些文章的链接，并且存储到Excel里，就像下面这个样子：



我们用Python。

## **环境**

要装Python，比较省事的办法是安装Anaconda套装。请到[这个网址](https://link.zhihu.com/?target=https%3A//www.anaconda.com/download/)下载Anaconda的最新版本。

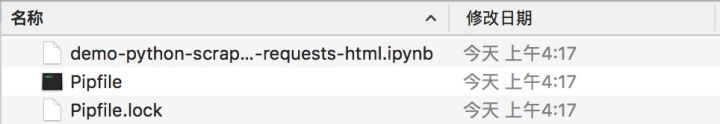


请选择左侧的 Python **3.6** 版本下载安装。

如果你需要具体的步骤指导，或者想知道Windows平台如何安装并运行Anaconda命令，请参考我为你准备的[视频教程](https://link.zhihu.com/?target=https%3A//www.jianshu.com/p/772740d57576)。

安装好Anaconda之后，请到[这个网址](https://link.zhihu.com/?target=https%3A//github.com/wshuyi/demo-python-scrape-webpage-with-requests-html/archive/master.zip)下载本教程配套的压缩包。

下载后解压，你会在生成的目录（下称“演示目录”）里面看到以下三个文件。



打开终端，用cd命令进入该**演示目录**。如果你不了解具体使用方法，也可以参考[视频教程](https://link.zhihu.com/?target=https%3A//www.jianshu.com/p/772740d57576)。

我们需要安装一些环境依赖包。

首先执行：

pip install pipenv

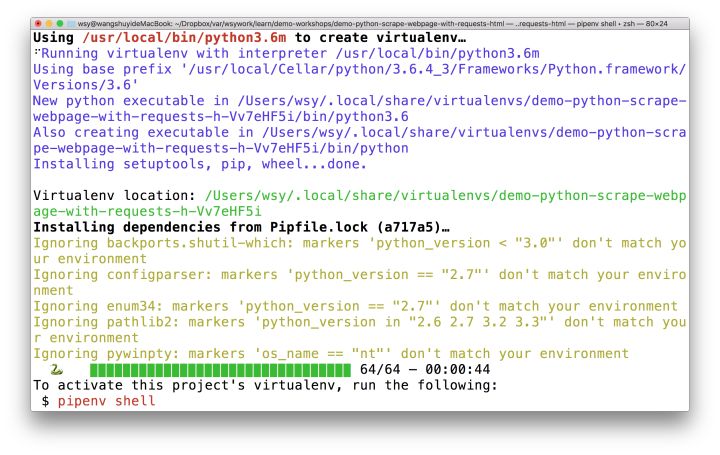
这里安装的，是一个优秀的 Python 软件包管理工具 pipenv 。

安装后，请执行：

pipenv install

看到演示目录下两个Pipfile开头的文件了吗？它们就是 pipenv 的设置文档。

pipenv 工具会依照它们，自动为我们安装所需要的全部依赖软件包。



上图里面有个绿色的进度条，提示所需安装软件数量和实际进度。

装好后，根据提示我们执行：

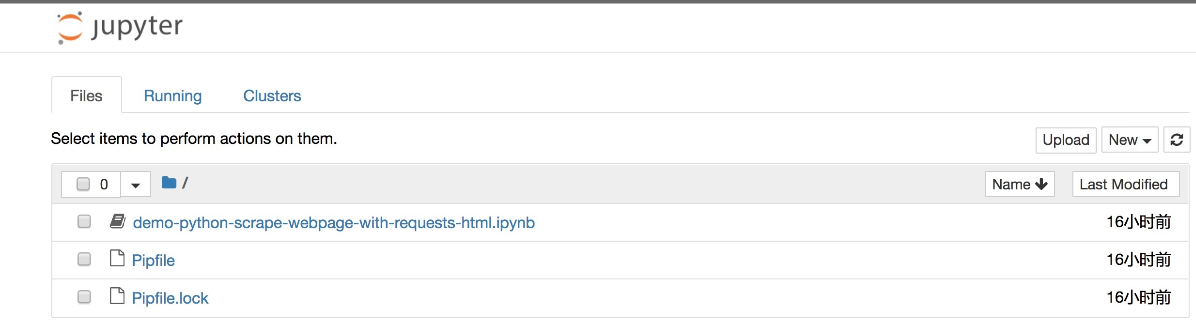
pipenv shell

此处请确认你的电脑上已经安装了 Google Chrome 浏览器。

我们执行：

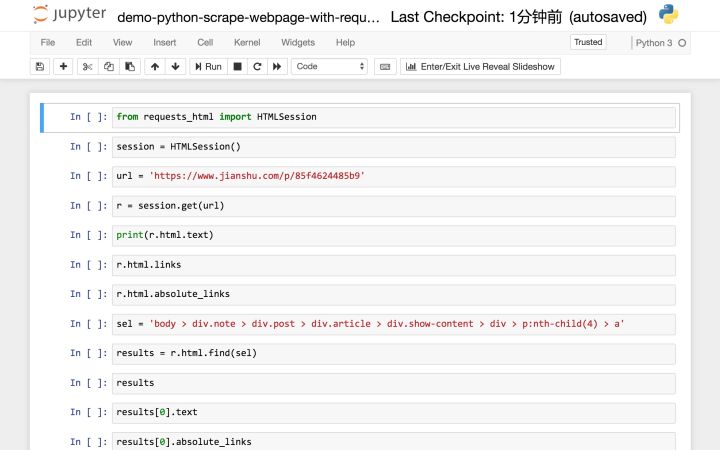
jupyter notebook

默认浏览器会开启，并启动 Jupyter 笔记本界面：

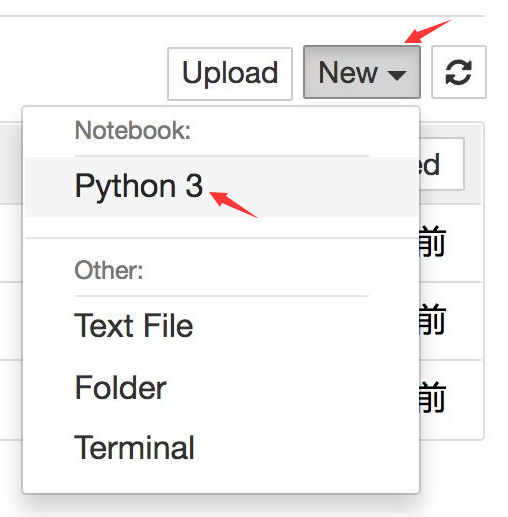


你可以直接点击文件列表中的第一项ipynb文件，可以看到本教程的全部示例代码。

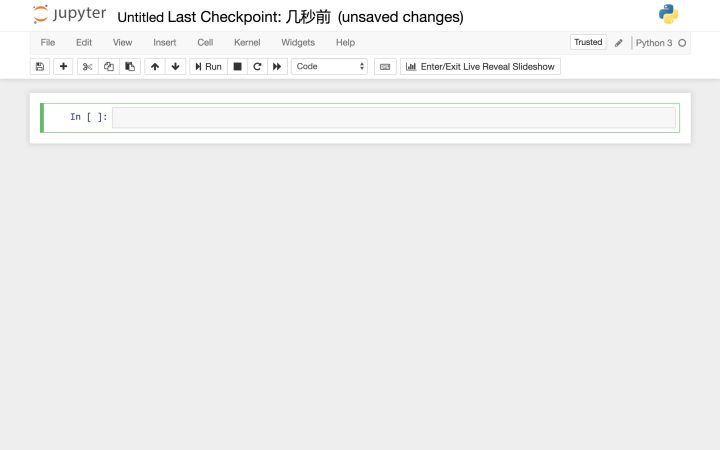
你可以一边看教程的讲解，一边依次执行这些代码。



但是，我**建议**的方法，是回到主界面下，新建一个新的空白 Python 3 笔记本。



请跟着教程，一个个字符输入相应的内容。这可以帮助你更为深刻地理解代码的含义，更高效地把技能内化。



准备工作结束，下面我们开始正式输入代码。

## **代码**

读入网页加以解析抓取，需要用到的软件包是 requests\_html 。我们此处并不需要这个软件包的全部功能，只读入其中的 HTMLSession 就可以。

from requests\_html import HTMLSession

然后，我们建立一个会话（session），即让Python作为一个客户端，和远端服务器交谈。

session = HTMLSession()

前面说了，我们打算采集信息的网页，是《[如何用《玉树芝兰》入门数据科学？](https://link.zhihu.com/?target=https%3A//www.jianshu.com/p/85f4624485b9)》一文。

我们找到它的网址，存储到url变量名中。

url = 'https://www.jianshu.com/p/85f4624485b9'

下面的语句，利用 session 的 get 功能，把这个链接对应的网页整个儿取回来。

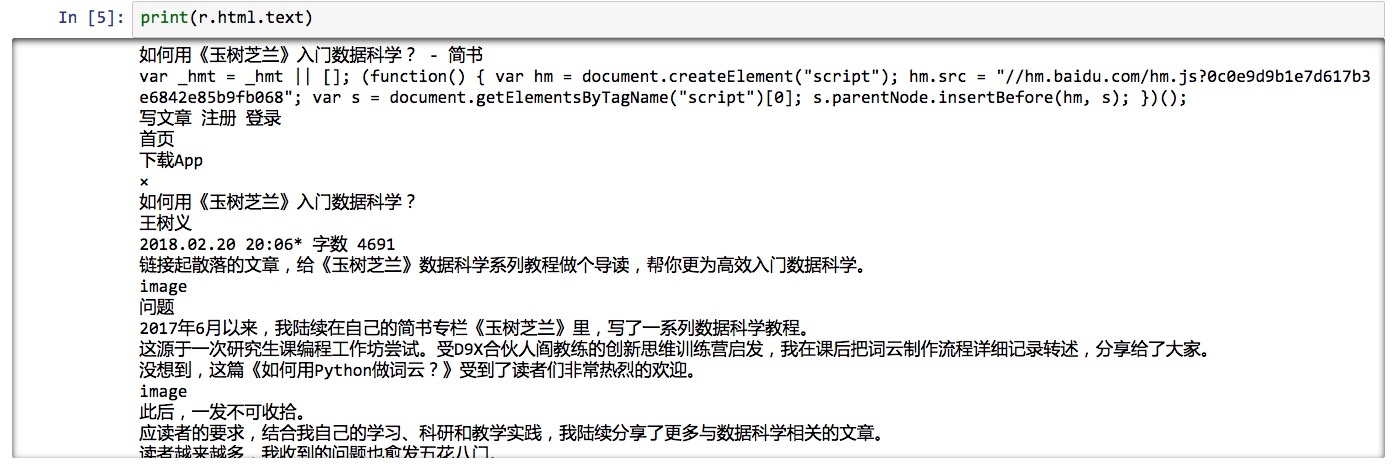
r = session.get(url)

网页里面都有什么内容呢？我们告诉Python，请把服务器传回来的内容当作HTML文件类型处理。我不想要看HTML里面那些乱七八糟的格式描述符，只看文字部分。

于是我们执行：

print(r.html.text)

获得的结果如下：



取回来的网页信息是正确的，内容是完整的。

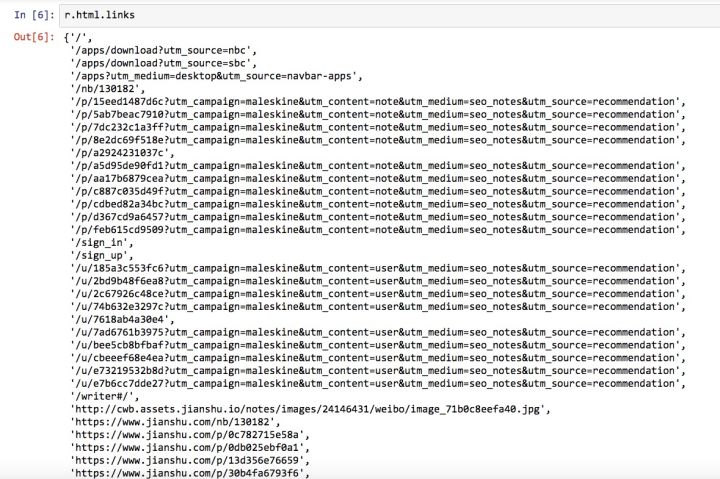
好了，我们来看看怎么趋近自己的目标吧。

我们先用**简单粗暴**的方法，尝试获得网页中包含的全部链接。

把返回的内容作为HTML文件类型，我们查看 links 属性：

r.html.links

返回的结果如下：



这么多链接啊！很兴奋吧？不过，你发现没有？这里许多链接，看似都不完全。

例如第一条结果，只有：

'/'

这是什么东西？是不是链接抓取错误啊？

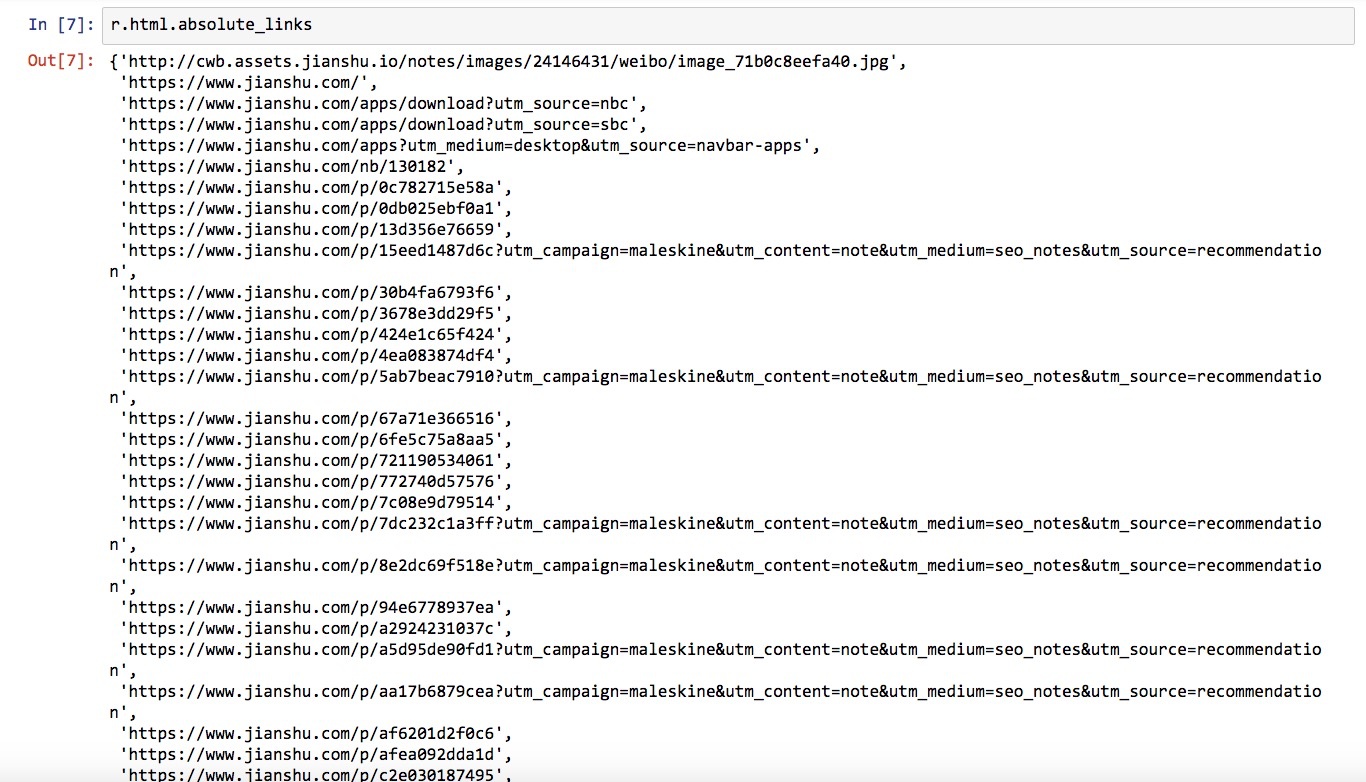
不是，这种看着不像链接的东西，叫做相对链接。它是某个链接，相对于我们采集的网页所在域名（<https://www.jianshu.com>）的路径。

这就好像我们在国内邮寄快递包裹，填单子的时候一般会写“XX省XX市……”，前面不需要加上国家名称。只有国际快递，才需要写上国名。

但是如果希望获得全部可以直接访问的带有域名的绝对链接，也只需要一条 Python 语句。

r.html.absolute\_links

这里，我们要的是“绝对”链接，于是我们就会获得下面的结果：



我们不光要找到链接，还得找到链接对应的描述文字呢，结果里包含吗？没有。

结果列表中的链接，都是我们需要的吗？不是。看长度，我们就能感觉出许多链接并不是文中描述其他数据科学文章的网址。这种简单粗暴直接罗列所有的HTML文件中所有链接的方法，对本任务行不通。那么我们该怎么办？要向Python 界定清楚我们要找的东西。这才是网页抓取的**关键**。

想想看，如果你想让助手（人类）帮你做这事儿，怎么办？你会告诉他：“寻找正文中全部可以点击的蓝色文字链接，拷贝文字到Excel表格，然后右键复制对应的链接，也拷贝到Excel表格。每个链接在Excel占一行，文字和链接各占一个单元格。”虽然这个操作执行起来麻烦，但是助手听懂后，就能帮你执行。同样的描述，你试试说给电脑听……不好意思，它不理解。因为你和助手看到的网页，是以下这个样子的。



但是电脑看到的网页，却是以下这个样子的，一团乱码。



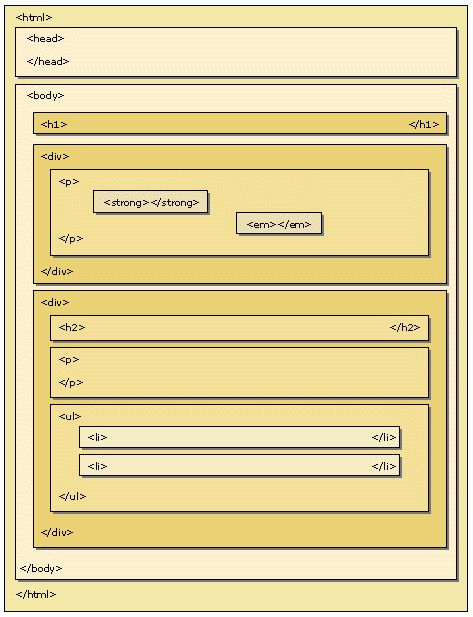
为了让你看得清楚源代码，浏览器还特意对不同类型的数据用了颜色区分，对行做了编号。

数据显示给电脑时，上述辅助可视功能是没有的。它只能看见一串串字符。那可怎么办？

仔细观察，你会发现这些HTML源代码里面，文字、图片链接内容前后，都会有一些被尖括号括起来的部分，这就叫做“标记”。

所谓HTML，就是一种标记语言（超文本标记语言，HyperText Markup Language）。

标记的作用是什么？它可以把整个的文件分解出结构和层次来。



（图片来源：[https://goo.gl/kWCqS6](https://link.zhihu.com/?target=https%3A//goo.gl/kWCqS6)）

如同你要发送包裹给某个人，可以按照“省-市-区-街道-小区-门牌”这样的结构来写地址，快递员也可以根据这个地址找到收件人。同样，我们对网页中某些特定内容感兴趣，可以依据这些标记的结构，顺藤摸瓜找出来。

这是不是意味着，你必须先学会HTML和CSS，才能进行网页内容抓取呢？

不是的，我们可以借助工具，帮你显著简化任务复杂度。这个工具，Google Chrome浏览器自带。

我们在样例文章页面上，点击鼠标右键，在出现的菜单里面选择“检查”。



这时，屏幕下方就会出现一个分栏。



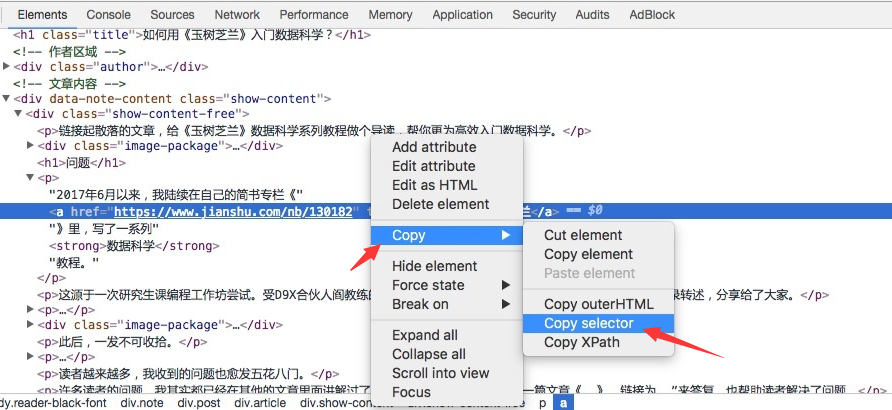
我们点击这个分栏左上角（上图红色标出）的按钮。然后把鼠标悬停在第一个文内链接（《玉树芝兰》）上面，点击一下。

￼

此时，你会发现下方分栏里面，内容也发生了变化。这个链接对应的源代码被放在分栏区域正中，高亮显示。



确认该区域就是我们要找的链接和文字描述后，我们鼠标右键选择高亮区域，并且在弹出的菜单中，选择 Copy -> Copy selector。



找一个文本编辑器，执行粘贴，就可以看见我们究竟复制下来了什么内容。

body > div.note > div.post > div.article > div.show-content > div > p:nth-child(4) > a

这一长串的标记，为电脑指出了：请你先找到 body 标记，进入它管辖的这个区域后去找 div.note 标记，然后找……最后找到 a 标记，这里就是要找的内容了。

回到Jupyter Notebook 中，用刚才获得的标记路径，定义变量sel。

sel = 'body > div.note > div.post > div.article > div.show-content > div > p:nth-child(4) > a'

让 Python 从返回内容中，查找 sel 对应的位置，把结果存到 results 变量中。

results = r.html.find(sel)

我们看看 results 里面都有什么。

results

这是结果：

[<Element 'a' href='https://www.jianshu.com/nb/130182' target='\_blank'>]

results 是个列表，只包含一项。这一项包含一个网址，就是我们要找的第一个链接（《玉树芝兰》）对应的网址。

可是文字描述“《玉树芝兰》”哪里去了？别着急，我们现在让 Python 显示 results 结果数据所对应的文本。

results[0].text

这是输出结果：

'玉树芝兰'

我们把链接也提取出来：

results[0].absolute\_links

显示的结果却是一个集合。

{'https://www.jianshu.com/nb/130182'}

我们不想要集合，只想要其中的链接字符串。所以我们先把它转换成列表，然后从中提取第一项，即网址链接。

list(results[0].absolute\_links)[0]

这次，终于获得我们想要的结果了：

'https://www.jianshu.com/nb/130182'

有了处理这第一个链接的经验，你信心大增，是吧？其他链接，也无非是找到标记路径，然后照猫画虎嘛。可是，如果每找一个链接，都需要手动输入上面这若干条语句，那也太麻烦了。

这里就是编程的技巧了。重复逐条运行的语句，如果工作顺利，我们就要尝试把它们归并起来，做个简单的函数。

对这个函数，只需给定一个选择路径（sel），它就把找到的所有描述文本和链接路径都返回给我们。

def get\_text\_link\_from\_sel(sel):

mylist = []

try:

results = r.html.find(sel)

for result in results:

mytext = result.text

mylink = list(result.absolute\_links)[0]

mylist.append((mytext, mylink))

return mylist

except:

return None

我们测试一下这个函数。还是用刚才的标记路径（sel）不变，试试看。

print(get\_text\_link\_from\_sel(sel))

输出结果如下：

[('玉树芝兰', 'https://www.jianshu.com/nb/130182')]

没问题，对吧？好，我们试试看第二个链接。

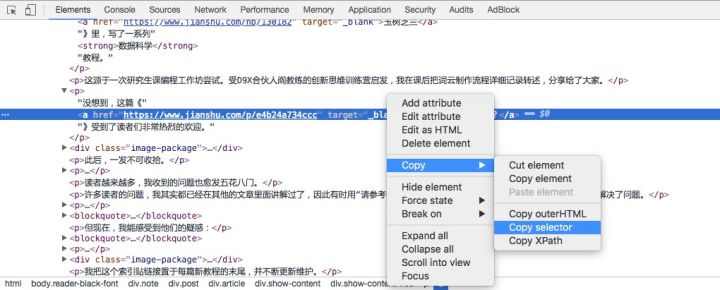
我们还是用刚才的方法，使用下面分栏左上角的按钮点击第二个链接。



下方出现的高亮内容就发生了变化：



我们还是用鼠标右键点击高亮部分，拷贝出 selector。



然后我们直接把获得的标记路径写到 Jupyter Notebook 里面。

sel = 'body > div.note > div.post > div.article > div.show-content > div > p:nth-child(6) > a'

用我们刚才编制的函数，看看输出结果是什么？

print(get\_text\_link\_from\_sel(sel))

输出如下：

[('如何用Python做词云？', 'https://www.jianshu.com/p/e4b24a734ccc')]

检验完毕，函数没有问题。

下一步做什么？你还打算去找第三个链接，仿照刚才的方法做？那你还不如全文手动摘取信息算了，更省事儿一些。我们要想办法把这个过程**自动化**。

对比一下刚刚两次我们找到的标记路径：

body > div.note > div.post > div.article > div.show-content > div > p:nth-child(4) > a

以及：

body > div.note > div.post > div.article > div.show-content > div > p:nth-child(6) > a

**发现规律**了没有？对，路径上其他的标记全都是一样的，唯独倒数第二个标记（"p"）后冒号后内容有区别。这就是我们自动化的关键了。

上述两个标记路径里面，因为指定了在第几个“子”(nth-child)文本段（paragraph,也就是"p"代表的含义）去找"a"这个标记，因此只返回来单一结果。

如果我们不限定"p"的具体位置信息呢？ 我们试试看，这次保留标记路径里面其他全部信息，只修改"p"这一点。

sel = 'body > div.note > div.post > div.article > div.show-content > div > p > a'

再次运行我们的函数：

print(get\_text\_link\_from\_sel(sel))

这是输出结果：



好了，我们要找的内容，全都在这儿了。但是，我们的工作还没完。我们还得把采集到的信息输出到Excel中保存起来。

还记得我们常用的数据框工具 Pandas 吗？又该让它大显神通了。

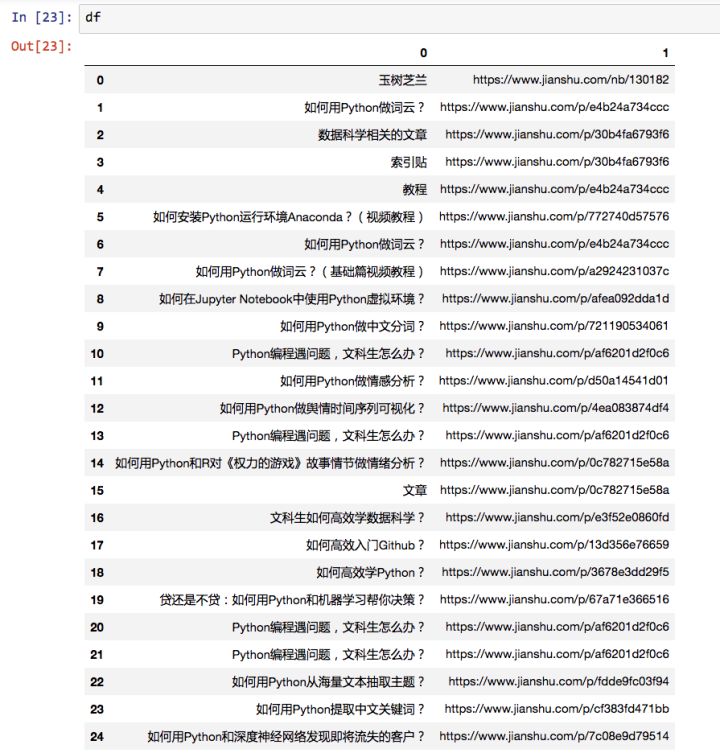
import pandas as pd

只需要这一行命令，就能把刚才的列表变成数据框（DataFrame）（带上表头和行号）：

df = pd.DataFrame(get\_text\_link\_from\_sel(sel))

看看数据框内容：

df



内容没问题，不过我们对表头不大满意，将表头0，1的数字分别换为更有意义的文字名称“text”和“link”,使读者容易明白列的含义。

df.columns = ['text', 'link']

再看看数据框，表头内容变成了文字的列标题：

df



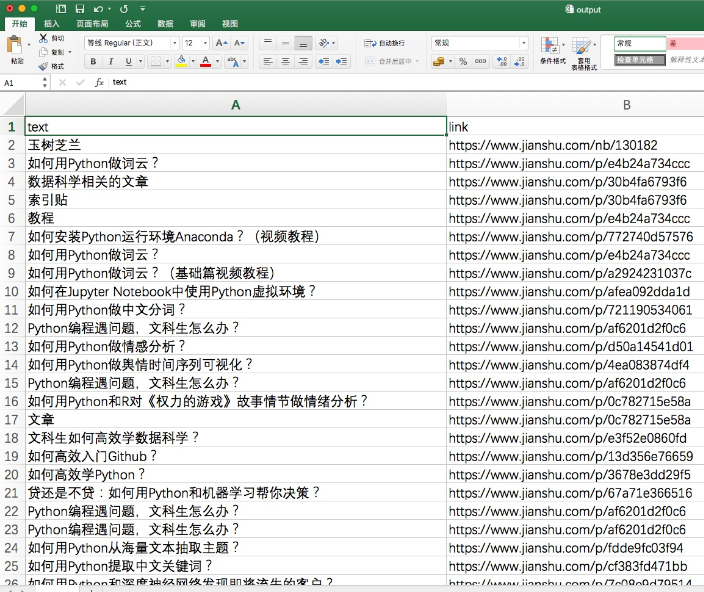
好了，下面就可以把抓取的内容输出到Excel中了。

Pandas内置的命令df.to\_csv 能把数据框变成csv格式，可以用Excel直接打开查看。

df.to\_csv('output.csv', encoding='gbk', index=False)

注意这里需要指定encoding（编码）为GBK，否则默认的utf-8编码在Excel中查看的时候，有可能是乱码。

我们看看最终生成的csv文件吧。



很有成就感，是不是？

## **小结**

本文为你展示了用Python自动网页抓取的基础技能。

希望阅读并动手实践后，你能掌握以下知识点：

* 网页抓取与网络爬虫之间的联系与区别；
* 如何用 pipenv 快速构建指定的 Python 开发环境，自动安装好依赖软件包；
* 如何用 Google Chrome 的内置检查功能，快速定位感兴趣内容的标记路径；
* 如何用 requests-html 包来解析网页，查询获得需要的内容元素；
* 如何用 Pandas 数据框工具整理数据，并且输出到 Excel。

或许，你觉得这篇文章过于浅白，不能满足你的要求。文中只展示了如何从一个网页抓取信息，可你要处理的网页成千上万啊。别着急。本质上说，抓取一个网页，和抓取10000个网页，在流程上是一样的。而且，从咱们的例子里，你是不是已经尝试了抓取链接？

有了链接作为基础，你就可以滚雪球，让Python爬虫“爬”到解析出来的链接上，做进一步的处理。

将来，你可能还要应对实践场景中的一些棘手问题：

* 如何把抓取的功能扩展到某一范内内的所有网页？
* 如何爬取Javascript动态网页？
* 假设你爬取的网站对每个IP的访问频率做出限定，怎么办？
* ……

这些问题的解决办法，我希望在今后的教程里面，一一和你分享。

需要注意的是，网络爬虫抓取数据，虽然功能强大，但学习与实践起来有一定门槛。

当你面临数据获取任务时，应该先检查一下这个清单：

* 有没有别人已经整理好的数据集合可以直接下载？
* 网站有没有对你需要的数据提供API访问与获取方式？
* 有没有人针对你的需求，编好了定制爬虫，供你直接调用？

如果答案是都没有，才需要你自己编写脚本，调动爬虫来抓取。

为了巩固学习的知识，请你换一个其他网页，以咱们的代码作为基础修改后，抓取其中你感兴趣的内容。

如果能把你抓取的过程记录下来，在评论区将记录链接分享给大家，就更好了。

因为**刻意练习**是掌握实践技能的最好方式，而**教是最好的学**。

本文主要内容讲解完毕。这里给你提一个疑问，供你思考：

我们解析并且存储的链接，其实是有重复的：



这并不是我们的代码有误，而是在《[如何用《玉树芝兰》入门数据科学？](https://link.zhihu.com/?target=https%3A//www.jianshu.com/p/85f4624485b9)》一文里，本来就多次引用过一些文章，所以重复的链接就都被抓取出来了。但是你存储的时候，也许不希望保留重复链接。这种情况下，你该如何修改代码，才能保证抓取和保存的链接没有重复呢？

## **讨论**

你对Python爬虫感兴趣吗？在哪些数据采集任务上使用过它？有没有其他更高效的方式，来达成数据采集目的？欢迎留言，把你的经验和思考分享给大家，我们一起交流讨论。

如果你对我的文章感兴趣，欢迎点赞，并且微信关注和置顶我的公众号“玉树芝兰”(nkwangshuyi)。

如果本文可能对你身边的亲友有帮助，也欢迎你把本文通过微博或朋友圈分享给他们。让他们一起参与到我们的讨论中来。

编辑于 2018-03-03

# Python3.6有关beautifulsoup成功安装的经验

原文链接：<https://blog.csdn.net/llx1026/article/details/77477289>

原创llx1026 最后发布于2017-08-22 10:46:08 阅读数 1399 收藏

今天在练习某公众号的用python爬取战狼影评的小练习，本人就尝试着练习，发现有一个BeaatifulSoup的库需要安装，我的是python3.6版本，尝试了网上好多方法均不成功，错误提示是ImportError: cannot import name 'HTMLParseError'，百思不得其解，看了一篇博客讲的是MAC系统的beautifulsoup安装，抱着试试看的心态，在windows下本人也安装成功了。

打开电脑终端，即在开始菜单->搜索程序和文件处输入cmd,  按回车键，出来黑色终端显示器只需输入：

pip3 install --upgrade beautifulsoup4  注意：不要用pip，要不然安装不成功，要用pip3才可以

然后就ok了，附原参考博客<http://blog.csdn.net/u010358168/article/details/62040603>

# Python3 爬虫配置BeautifulSoup4+lxml+html5lib

原创逆風的薔薇 最后发布于2016-06-01 10:08:07 阅读数 18514 收藏

展开

序

Beautiful Soup 是一个可以从HTML或XML文件中提取数据的Python库.它能够通过你喜欢的转换器实现惯用的文档导航,查找,修改文档的方式.Beautiful Soup会帮你节省数小时甚至数天的工作时间。

Beautiful Soup支持Python标准库中的HTML解析器,还支持一些第三方的解析器,其中一个是 lxml 。

另一个可供选择的解析器是纯Python实现的 html5lib , html5lib的解析方式与浏览器相同。

Windows平台 + Python3.5

安装BeautifulSoup4

方法一：打开cmd，运行pip install BeautifulSoup4

如上图所示，由于我已经安装过了。可以使用 --upgrade来升级为最新版本。

方法二：去官网BeautifulSoup4源码下载 -- 戳我吧！下载源码，编译运行。

至此，便安装完毕。

验证成功，编译一个.py文件，输入from bs4 import BeautifulSoup4，不会报错即代表安装成功。

安装html5lib

第二步，我们安装网页文件解析器htm5lib，只需直接运行pip install html5lib即可：

安装lxml

在Windows下，安装lxml费了一点劲儿，不能直接通过命令成功安装。我们需要去官方网站下载与平台完全一致的版本，手动安装。

首先，查看我们的平台依赖的工具版本：

然后，去官网下载对应的.whl文件。lxml 官方下载链接，请猛戳我~~~

Ctrl + F，输入lxml，找到下面这段：

Lxml, a binding for the libxml2 and libxslt libraries.

lxml‑3.4.4‑cp27‑none‑win32.whl

lxml‑3.4.4‑cp27‑none‑win\_amd64.whl

lxml‑3.4.4‑cp33‑none‑win32.whl

lxml‑3.4.4‑cp33‑none‑win\_amd64.whl

lxml‑3.4.4‑cp34‑none‑win32.whl

lxml‑3.4.4‑cp34‑none‑win\_amd64.whl

lxml‑3.4.4‑cp35‑none‑win32.whl

lxml‑3.4.4‑cp35‑none‑win\_amd64.whl

cp后面是Python的版本号，27表示2.7，根据你的Python版本选择下载。

最后进行安装，打开cmd，先运行pip install wheel安装wheel工具，做好准备工作。

接着运行pip install \*.whl文件，我的对应版本为lxml-3.6.0-cp35-cp35m-win\_amd64.whl即可成功安装lxml解析器。

至此，三个工具都安装完毕。

对于Linux平台下，安装就很简单了，直接利用三个命令即可完成：

pip install BeautifulSoup4 或 easy\_install BeautifulSoup4

pip install html5lib

pip install lxml

使用BeautifulSoup

我们编辑一段html文档，利用BeautifulSoup库进行解析：

html = """

<html><head><title>The Dormouse's story</title></head>

<body>

<p class="title"><b>The Dormouse's story</b></p>

<p class="story">Once upon a time there were three little sisters; and their names were

<a href="http://example.com/elsie" class="sister" id="link1">Elsie</a>,

<a href="http://example.com/lacie" class="sister" id="link2">Lacie</a> and

<a href="http://example.com/tillie" class="sister" id="link3">Tillie</a>;

and they lived at the bottom of a well.</p>

<p class="story">...</p>

"""

from bs4 import BeautifulSoup

#添加一个解析器

soup = BeautifulSoup(html,'html5lib')

print(soup.title)

print(soup.title.name)

print(soup.title.text)

print(soup.body)

#从文档中找到所有<a>标签的内容

for link in soup.find\_all('a'):

print(link.get('href'))

#从文档中找到所有文字内容

print(soup.get\_text())

注意：

在声明BeautifulSoup对象的时候要明确解析器 soup = BeautifulSoup(html,'html5lib')，否则写为 soup = BeautifulSoup(html) 会有警告。

运行上述代码：

我们发现，BeautifulSoup可以十分方便的提取Html的结构化数据。这就为我们解析网页文件内容，爬取目标元素提供了极大的帮助。

这只是一个小小的例子，BeautifulSoup库的功能十分强大，赶紧去官方文档学习吧~

BeautifulSoup4官方文档在这里~~~

# 详解Python 用 requests + Beautiful Soup 爬取房天下

[yuanzhitang](https://blog.51cto.com/yuanzhitang)关注0人评论[15917人阅读2018-01-07 11:44:30](javascript:;) <https://blog.51cto.com/yuanzhitang/2058281>

最近一直在关注Python写爬虫相关的知识，尝试了采用requests + Beautiful Soup来爬取房天下（原搜房网）的推荐新楼盘。不用不知道，一用发现有惊喜也有惊吓，本文就一同记录下惊喜和踩的一些乱码的坑。

首先，觉得Beautiful soup解析网页更加符合人类的常规思维，比使用正则表达式（python中的re库）更容易理解。 同时关于requests遇到了中文字符和特殊字符解码的问题。本文都将给于深入的解说。

**软件环境**

Python    : 3.6.0

PyCharm: Community 2017.2

库1 : requests

库2 : beautifulsoup

**Requests知识概要**

Requests 是用Python语言编写，基于 urllib，采用 Apache2 Licensed 开源协议的 HTTP 库。它比 urllib 更加方便，可以节约我们大量的工作，完全满足 HTTP 测试需求。

完全支持Python3哦

1. 安装requests

   可以采用pip安装requests,具体代码是：

pip install requests

2. 简单例子

   下面例子中是几个常用的属性，response.text可以得到html的源代码，response.encoding默认为ISO-8859-1，此编码针对英文没有问题，如果我们的网页中带有中文，将会引起乱码的问题。后面有解决方案。

import requests

response = requests.get("http://sh.fang.com/") #获取一个Http请求

print(response.encoding) #当前编码，默认为 ISO-8859-1

print(response.apparent\_encoding) #当前网页的内容的实际编码

print(response.content) #content返回的是bytes型的原始数据

print(response.text) #text返回的是处理过的Unicode型的数据

**Beautiful Soup简要知识**

1.Beautiful Soup 是一个非常流行的 Python 模块。 该模块可以解析网页， 并

提供定位 内 容的便捷接 口 。 如果你还没有安装该模块， 可以使用下面的命令

安装其最新版本：

pip install beautifulsoup4

使用 Beautiful Soup 的第一步是将下载的 HTML 内容解析为 soup 文档 。

2. 如何引用Beautiful Soup

from bs4 import BeautifulSoup

3. 一个简单例子

from bs4 import BeautifulSoup

html = "<ul class='country'><li>中国</li><li>美国</li></ul>"

# 解析html然后得到一个soup文档

soup = BeautifulSoup(html,'html.parser')

html = soup.prettify()

ul = soup.find('ul',attrs={'class':'country'})

#下面代码只返回一个匹配的元素

print(ul.find('li')) # returns the first match element

#下面代码返回又有匹配的元素

for countryname in ul.find\_all('li'):

print(countryname)

**实战爬取房天下推荐新楼盘**

1. **在Chrome中打开sh.fang.com地址，按F12观察新盘推荐tab的网页源代码结构**

主要结构是: 顶层为一个名为ti011的div，下面有四个class为tenrtd的div用于四个楼盘的现实。每个楼盘下面有一个class = "text1"的div存储了楼盘名称，另一个class = "text2"的存储了楼盘的价格。



**2. 第一版代码完成如下，但是发现有一个中文乱码的问题**

from bs4 import BeautifulSoup

import requests

rep = requests.get("http://sh.fang.com/")

html = rep.text

soup = BeautifulSoup(html, 'html.parser')

#获取顶层 新盘推荐 的整个div

div = soup.find('div',attrs={'id':'ti011'})

#获取四个楼盘的div，更具他们的class = "tenrtd"

for house in div.find\_all('div', attrs={'class': 'tenrtd'}):

# 根据class="text1"获取存储楼盘标题的div

titleDiv = house.find('div', attrs={'class': 'text1'})

title = titleDiv.find('a').text

# 根据class="text2"获取存储楼盘价格的div

priceDiv = house.find('div', attrs={'class': 'text2'})

price = priceDiv.find('b').text

print(title, " ", price)

输出的结果尽然是这样， Why？

 ÖÐ½ðº£ÌÄÍå     48000Ôª/©O

 »ªÒêÒÝÆ·À½Íå     43057Ôª/©O

 ÖÐ¹ú¹é¹È´´¿ÍSOHO     ¼Û¸ñ´ý¶¨

 ÐÂÎ÷ÌÁ¿×È¸³Ç     14000Ôª/©O

**3. 研究中文乱码问题**

    中文乱码也算是requests常见的一个问题，为什么会这样的呢，看bs自己的文档描述

Encodings

When you receive a response, Requests makes a guess at the encoding to use for decoding the response

when you access the Response.text attribute. Requests will first check for an encoding in the HTTP

header, and if none is present, will use chardet to attempt to guess the encoding.

The only time Requests will not do this is if no explicit charset is present in the HTTP headersand

the Content-Type header contains text. In this situation, RFC 2616 specifies that the default charset

must be ISO-8859-1. Requests follows the specification in this case. If you require a different encoding,

you can manually set the Response.encoding property, or use the rawResponse.content.

    其实重要的也就是如果没有在requests的header部分里面指定encoding的话，那么默认采用ISO-8859-1这个编码去解码response.content里面的字节数据。ISO-8859-1针对英文是没有问题的，但是针对中文就不行了。

    解决思路当然就是换encoding，但是我们应该用什么编码呢？下面这个代码可以看到网页的实际encoding，下面代码将输出：GB2312 ，所以我们采用GB2312来解码我们的数据。

print(rep.apparent\_encoding)

**4.换编码解决中文字符**

    经过上面的研究，我们修订代码对response设定encoding = "GB2312"

rep.encoding = "GB2312"

   运行后结果如下：

  中金海棠湾     48000元/�O

 华谊逸品澜湾     43057元/�O

 中国归谷创客SOHO     价格待定

 新西塘孔雀城     14000元/�O

   发现代码显示的结果中㎡又乱码了，不应该啊，这又是为什么呢？ 下面接着研究。。。

**5. 研究解决特殊字符乱码问题**

       引起乱码的原因估计就是在字符集中找不到特定的字符，比如这个㎡。是不是GB2312这个字符集不够全面呢？带着这个疑问去查阅相关的资料关于中文的几个编码：

* GB2312

GB 2312 或 GB 2312-80 是中国国家标准简体中文字符集，全称《信息交换用汉字编码字符集·基本集》，又称 GB 0，由中国国家标准总局发布，1981 年 5 月 1 日实施。GB 2312 编码通行于中国大陆；新加坡等地也采用此编码。中国大陆几乎所有的中文系统和国际化的软件都支持 GB 2312。**GB 2312 标准共收录 6763 个汉字**，其中一级汉字 3755 个，二级汉字 3008 个；同时收录了包括拉丁字母、希腊字母、日文平假名及片假名字母、俄语西里尔字母在内的 682 个字符。GB 2312 的出现，基本满足了汉字的计算机处理需要，它所收录的汉字已经覆盖中国大陆99.75% 的使用频率。**对于人名、古汉语等方面出现的罕用字，GB 2312 不能处理，这导致了后来 GBK 及 GB 18030 汉字字符集的出现**。

* GBK

GBK 即汉字内码扩展规范，K 为汉语拼音 Kuo Zhan（扩展）中“扩”字的声母。英文全称 Chinese Internal Code Specification。GBK 共收入 21886 个汉字和图形符号，包括：GB 2312 中的全部汉字、非汉字符号。BIG5 中的全部汉字。与 ISO 10646 相应的国家标准 GB 13000 中的其它 CJK 汉字，以上合计 20902 个汉字。其它汉字、部首、符号，共计 984 个。**GBK 向下与 GB 2312 完全兼容**，向上支持 ISO 10646 国际标准，在前者向后者过渡过程中起到的承上启下的作用。**GBK 采用双字节表示**，总体编码范围为 8140-FEFE 之间，首字节在 81-FE 之间，尾字节在 40-FE 之间，剔除 XX7F 一条线

* GB18030

GB 18030，全称：**国家标准 GB 18030-2005**《信息技术中文编码字符集》，是×××现时最新的内码字集，是 GB 18030-2000《信息技术信息交换用汉字编码字符集基本集的扩充》的修订版。GB 18030 与 GB 2312-1980 和 GBK 兼容，**共收录汉字70244个**。与 UTF-8 相同，采用多字节编码，每个字可以由 1 个、2 个或 4 个字节组成。编码空间庞大，最多可定义 161 万个字符。支持中国国内少数民族的文字，不需要动用造字区。汉字收录范围包含繁体汉字以及日韩汉字

从上文中可以简要的得出**GB 2312 过时标准、GBK 微软标准、GB 18030 国家标准，**

**字符个数方面：GB 18030 > GBK > GB2312**

**所以我们决定采用 GB18030来解码我们的数据，代码改动如下：**

rep.encoding = "gb18030"

**附上完整的代码**

from bs4 import BeautifulSoup

import requests

rep = requests.get("http://sh.fang.com/")

rep.encoding = "gb18030"

html = rep.text

soup = BeautifulSoup(html, 'html.parser')

#获取顶层 新盘推荐 的整个div

div = soup.find('div',attrs={'id':'ti011'})

#获取四个楼盘的div，更具他们的class = "tenrtd"

for house in div.find\_all('div', attrs={'class': 'tenrtd'}):

# 根据class="text1"获取存储楼盘标题的div

titleDiv = house.find('div', attrs={'class': 'text1'})

title = titleDiv.find('a').text

# 根据class="text2"获取存储楼盘价格的div

priceDiv = house.find('div', attrs={'class': 'text2'})

price = priceDiv.find('b').text

print(title, " ", price)

输出结果：

 中金海棠湾     48000元/㎡

 华谊逸品澜湾     43057元/㎡

 中国归谷创客SOHO     价格待定

 新西塘孔雀城     14000元/㎡

问题解决了，关键知识点总结：

1. 可以采用requests库来获取网页html
2. 采用Beautiful soup基于html构建一个soup文档，然后用find 或者 find\_all方法查询自己需要的html节点
3. 根据目标网页的内容来更改response.encoding从而解决乱码问题

# 手把手教你如何用 Python 做情感分析

|  |  |
| --- | --- |
| 本文作者：[AI研习社](https://www.leiphone.com/author/AI%E7%A0%94%E4%B9%A0%E7%A4%BE394) | 2017-06-28 16:28 |

<https://www.leiphone.com/news/201706/YXVb0apveG0yYDeT.html>

商品评论挖掘、电影推荐、股市预测……情感分析大有用武之地。本文帮助你一步步用Python做出自己的情感分析结果，难道你不想试试看？

## **需求**

如果你关注数据科学研究或是商业实践，“情感分析”(sentiment analysis)这个词你应该不陌生吧？

维基百科上，情感分析的定义是：

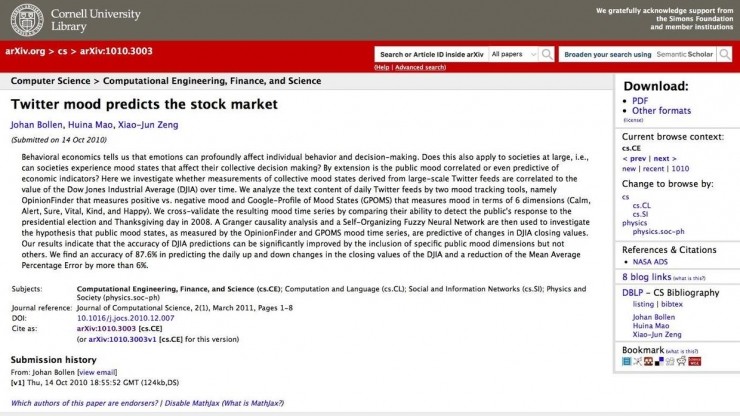
文本情感分析（也称为意见挖掘）是指用自然语言处理、文本挖掘以及计算机语言学等方法来识别和提取原素材中的主观信息。

听着很高大上，是吧？如果说得具体一点呢？

给你一段文本，你就可以用情感分析的自动化方法获得这一段内容里包含的情感色彩是什么。

神奇吧？

情感分析不是炫技工具。它是闷声发大财的方法。早在2010年，就有学者指出，可以依靠Twitter公开信息的情感分析来预测股市的涨落，准确率高达87.6%！



在这些学者看来，一旦你能够获得大量实时社交媒体文本数据，且利用情感分析的黑魔法，你就获得了一颗预测近期投资市场趋势的水晶球。这种用数据科学碾压竞争者的感受，是不是妙不可言啊？

大数据时代，我们可以获得的文本数据实在太多了。仅仅是大众点评、豆瓣和亚马逊上海量的评论信息就足够我们挥锹抡镐，深挖一通了。

你是不是疑惑，这么高深的技术，自己这个非计算机专业的文科生，如何才能应用呢？

不必担心。从前情感分析还只是实验室或者大公司的独门秘籍。现在早已飞入寻常百姓家。门槛的降低使得我们普通人也可以用Python的几行代码，完成大量文本的情感分析处理。

是不是摩拳擦掌，打算动手尝试了？那我们就开始吧。

## **安装**

为了更好地使用Python和相关软件包，你需要先安装Anaconda套装。详细的流程步骤请参考《 [如何用Python做词云](http://www.jianshu.com/p/e4b24a734ccc)》一文。

到你的系统“终端”(macOS, Linux)或者“命令提示符”(Windows)下，进入我们的工作目录demo，执行以下命令。

pip install snownlp

pip install -U textblob

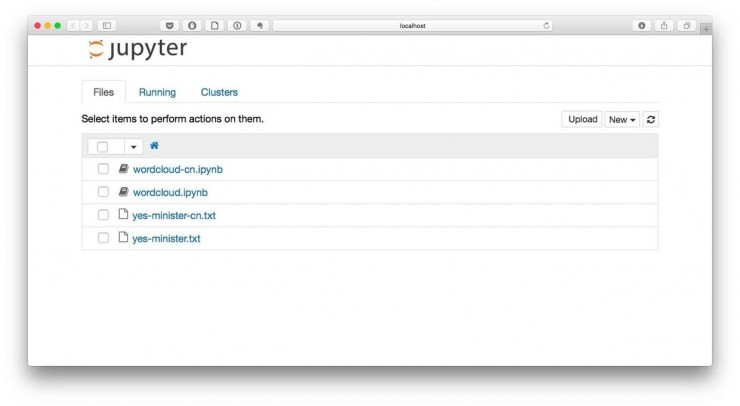
python -m textblob.download\_corpora

好了，至此你的情感分析运行环境已经配置完毕。

在终端或者命令提示符下键入：

jupyter notebook

你会看到目录里之前的那些文件，忽略他们就好。

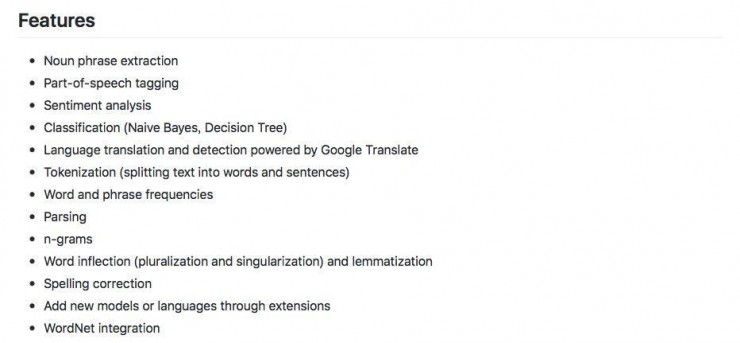


好了，下面我们就可以愉快地利用Python来编写程序，做文本情感分析了。

## **英文**

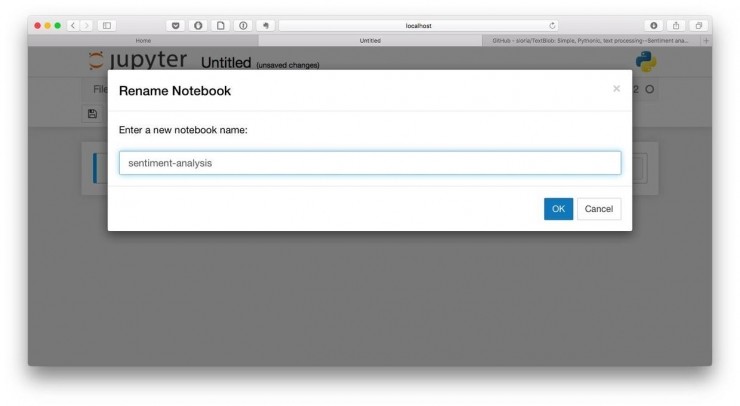
我们先来看英文文本的情感分析。

这里我们需要用到的是 TextBlob 包 ，相关链接为： <http://t.cn/RzFoHGR>



其实，从上图可以看出，这个包可以做许许多多跟文本处理相关的事情。本文我们只专注于情感分析这一项。其他功能以后有时间我们再介绍。

我们新建一个Python 2笔记本，并且将其命名为“sentiment-analysis”。



先准备一下英文文本数据。

text = "I am happy today. I feel sad today."

这里我们输入了两句话，把它存入了text这个变量里面。学了十几年英语的你，应该立即分辨出这两句话的情感属性。第一句是“我今天很高兴”，正面；第二句是“我今天很沮丧”，负面。

下面我们看看情感分析工具TextBlob能否正确识别这两句话的情感属性。

首先我们呼唤TextBlob出来。

from textblob import TextBlob

blob = TextBlob(text)

blob

按Shift+Enter执行，结果好像只是把这两句话原封不动打印了出来而已嘛。



别着急，TextBlob已经帮我们把一段文本分成了不同的句子。我们不妨看看它的划分对不对。

blob.sentences

执行后输出结果如下：

手把手教你如何用 Python 做情感分析

划分无误。可是你能断句有啥了不起？！我要情感分析结果！

你怎么这么着急啊？一步步来嘛。好，我们输出第一句的情感分析结果：

blob.sentences[0].sentiment

执行后，你会看到有意思的结果出现了：



情感极性0.8，主观性1.0。说明一下，情感极性的变化范围是[-1, 1]，-1代表完全负面，1代表完全正面。

既然我说自己“高兴”，那情感分析结果是正面的就对了啊。

趁热打铁，我们看第二句。

blob.sentences[1].sentiment

执行后结果如下：

手把手教你如何用 Python 做情感分析

“沮丧”对应的情感极性是负的0.5，没毛病！

更有趣的是，我们还可以让TextBlob综合分析出整段文本的情感。

blob.sentiment

执行结果是什么？ 给你10秒钟，猜猜看。不卖关子了，是这样的：

手把手教你如何用 Python 做情感分析

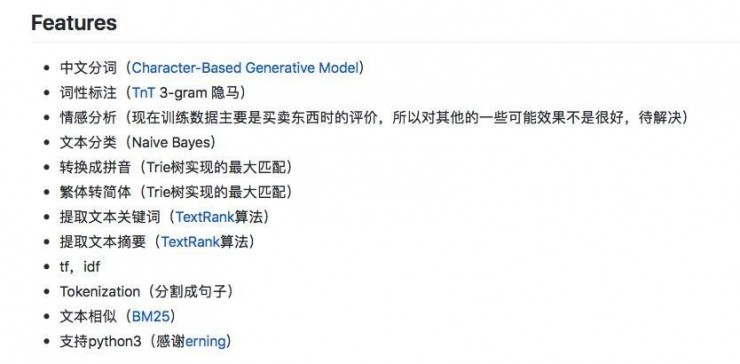
你可能会觉得没有道理。怎么一句“高兴”，一句“沮丧”，合并起来最后会得到正向结果呢？

首先不同极性的词，在数值上是有区别的。我们应该可以找到比“沮丧”更为负面的词汇。而且这也符合逻辑，谁会这么“天上一脚，地下一脚”矛盾地描述自己此时的心情呢？

## **中文**

试验了英文文本情感分析，我们该回归母语了。毕竟，互联网上我们平时接触最多的文本，还是中文的。

中文文本分析，我们使用的是 SnowNLP 包 。这个包跟TextBlob一样，也是多才多艺的。



SnowNLP 的相关链接如下：

<http://t.cn/8kf1c3p>

我们还是先准备一下文本。这次我们换2个形容词试试看。

text = u"我今天很快乐。我今天很愤怒。"

注意在引号前面我们加了一个字母u，它很重要。因为它提示Python，“这一段我们输入的文本编码格式是Unicode，别搞错了哦”。至于文本编码格式的细节，有机会我们再详细聊。

好了，文本有了，下面我们让SnowNLP来工作吧。

from snownlp import SnowNLP

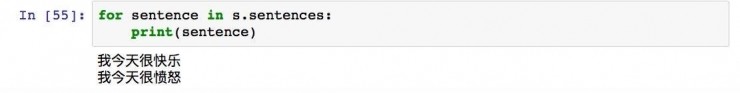
s = SnowNLP(text)

我们想看看SnowNLP能不能像TextBlob一样正确划分我们输入的句子，所以我们执行以下输出：

for sentence in s.sentences:

 print(sentence)

执行的结果是这样的：



好的，看来SnowNLP对句子的划分是正确的。

我们来看第一句的情感分析结果吧。

s1 = SnowNLP(s.sentences[0])

s1.sentiments

执行后的结果是：



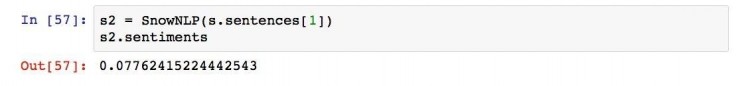
看来“快乐”这个关键词真是很能说明问题。基本上得到满分了。

我们来看第二句：

s2 = SnowNLP(s.sentences[1])

s2.sentiments

执行结果如下：



这里你肯定发现了问题——“愤怒”这个词表达了如此强烈的负面情感，为何得分依然是正的？

这是因为SnowNLP和textblob的计分方法不同。SnowNLP的情感分析取值，表达的是“这句话代表正面情感的概率”。也就是说，对“我今天很愤怒”一句，SnowNLP认为，它表达正面情感的概率很低很低。

这么解释就合理多了。

## 小结

学会了基本招式，很开心吧？下面你可以自己找一些中英文文本来实践情感分析了。

但是你可能很快就会遇到问题。例如你输入一些明确的负面情绪语句，得到的结果却很正面。

不要以为自己又被忽悠了。我来解释一下问题出在哪儿。

首先，许多语句的情感判定需要上下文和背景知识，因此如果这类信息缺乏，判别正确率就会受到影响。这就是人比机器（至少在目前）更强大的地方。

其次，任何一个情感分析工具，实际上都是被训练出来的。训练时用的是什么文本材料，直接影响到模型的适应性。

例如SnowNLP，它的训练文本就是评论数据。因此，你如果用它来分析中文评论信息，效果应该不错。但是，如果你用它分析其他类型的文本——例如小说、诗歌等，效果就会大打折扣。因为这样的文本数据组合方式，它之前没有见过。

解决办法当然有，就是用其他类型的文本去训练它。见多识广，自然就“见惯不怪”了。至于该如何训练，请和相关软件包的作者联系咨询。

## 讨论

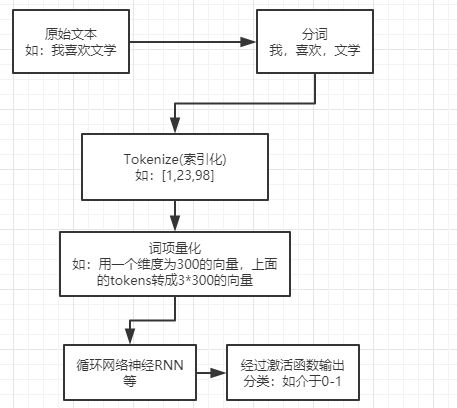
除了本文提到的文本分析应用领域，你还知道哪些其他的工作可以用情感分析来自动化辅助完成？除TextBlob和SnowNLP外，你还知道哪些开放免费软件包可以帮助我们完成情感分析工作？欢迎留言分享给大家，我们一起交流讨论

# Espresso-[使用TensorFlow进行中文自然语言处理的情感分析](https://www.cnblogs.com/zhuxiang1633/p/10331618.html)（笔记）

**对应的视频见B站和Youtube，有相应的代码压缩包**

### 1 TensorFlow使用

分析流程：



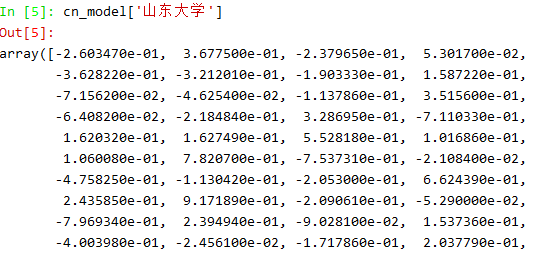
#### 1.1  使用gensim加载预训练中文分词embedding

 加载预训练词向量模型：https://github.com/Embedding/Chinese-Word-Vectors/

from gensim.models import KeyedVectors

cn\_model = KeyedVectors.load\_word2vec\_format('H:/词向量/word+Ngram/sgns.zhihu.bigram', binary=False)

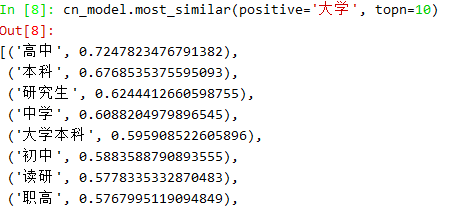
查看词语的向量模型表示： 维度为300



 词语相似度：向量余弦值

https://img2018.cnblogs.com/blog/1250855/201901/1250855-20190128175847609-472214808.png

最相似的词：



#### 1.2 训练样本预料

　　准备一个训练集，4000个酒店评论，其中2000条为pos积极的，2000条为消极的，每条评论放在一个文件中。

 1）文本预处理，分词、索引化

读取数据

[复制代码](javascript:void(0);)

import os

import re

import jieba

from gensim.models import KeyedVectors

cn\_model = KeyedVectors.load\_word2vec\_format('H:/word+Ngram/sgns.zhihu.bigram', binary=False)

baseDir = "H:/谭松波老师8++酒店评论++语料/utf-8/4000"

pos\_txts = os.listdir("H:/谭松波老师8++酒店评论++语料/utf-8/4000/pos")

neg\_txts = os.listdir("H:/bishe/NLP/训练集/谭松波老师8++酒店评论++语料/utf-8/4000/neg")

train\_text\_orig = []

for i in range(len(pos\_txts)):

with open(baseDir+"/pos/"+pos\_txts[i], errors="ignore", encoding="utf-8") as f:

text = f.read().strip()

train\_text\_orig.append(text)

f.close()

for i in range(len(neg\_txts)):

with open(baseDir+"/neg/"+neg\_txts[i], errors="ignore", encoding="utf-8") as f:

text = f.read().strip()

train\_text\_orig.append(text)

f.close()

[复制代码](javascript:void(0);)

分词，建立索引：

[复制代码](javascript:void(0);)

# [[句子词索引],[]]

train\_tokens = []

for text in train\_text\_orig:

# 去掉标点符号

text = re.sub("[\s+\.\!\/\_,$%^\*(+\"\')]+|[+——()?【】“”！，。？、~@#￥%……&\*（）]+", "", text)

# 结巴分词

cut = jieba.cut(text)

# 结巴分词结果为一个生成器

cut\_list = [i for i in cut]

for i, word in enumerate(cut\_list):

try:

# 将词转换成索引

cut\_list[i] = cn\_model.vocab[word].index

except KeyError:

cut\_list[i] = 0

train\_tokens.append(cut\_list)

[复制代码](javascript:void(0);)

2）文本长度标准化：

长度参差不齐，我们需要将长度标准化，方便模型进行训练，如果长度太短，会损失太多的信息，而长度太长会浪费太多计算资源  
所以说我们要取一个这种的方案，让这个长度基本上涵盖所有的训练样本，又不损失太多的信息

样本长度分布图：

[复制代码](javascript:void(0);)

# 看一下样本长度分布图

import matplotlib.pyplot as plt

plt.hist(num\_tokens, bins=100)

plt.xlim(0, 400)

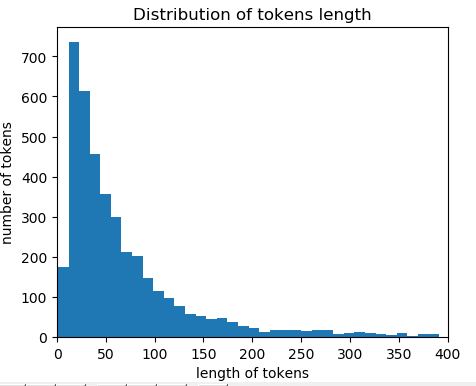
plt.ylabel("number of tokens")

plt.xlabel("length of tokens")

plt.title("Distribution of tokens length")

plt.show()

[复制代码](javascript:void(0);)



# 取tokens平均值加上两个tokens的标准差

# 假设tokens长度的分布符合正太分布，则max\_tokens这个值可以涵盖95%左右的样本

max\_tokens = np.mean(num\_tokens) + 2 \* np.std(num\_tokens)

# 取tokens的长度为236时，大约95%左右的样本被涵盖

# 对于长度不足的进行padding，过长的进行修剪

np.sum(num\_tokens<max\_tokens)/len(num\_tokens)

**反tokenize**

[复制代码](javascript:void(0);)

def reverse\_token(tokens):

'''

将索引化的句子还原

:param tokens: 句子 [词语，..]

:return:

'''

text = ""

for i in tokens:

if i!=0:

text = text+cn\_model.index2word[i]

else:

text =text+" "

return text

[复制代码](javascript:void(0);)

**3）准备Emdedding Matrix（词向量矩阵）**

    根据Keras的要求，我们需要准一个维度为(numwords, embeddingdim)的矩阵，num words代表我们使用的词汇的数量，emdedding dimension在我们预训练词向量模型中是300，每个词汇都用长度为300的向量表示（例如: 较好 ->[  0.056964, -0.127308, -0.118041,...]），注意词向量矩阵是作为训练模型的工具，

[复制代码](javascript:void(0);)

# 初始化词向量矩阵-embedding matrix（只用前50000个词）

num\_words = 50000

embedding\_matrix = np.zeros((num\_words, embedding\_dim))

# 维度为(50000, 300)的矩阵

for i in range(num\_words):

embedding\_matrix[i,:]=cn\_model[cn\_model.index2word[i]] # 将词向量赋值到词向量矩阵中

embedding\_matrix = embedding\_matrix.astype("float32")

# 检查赋值是否正确

np.sum(cn\_model[cn\_model.index2word[333]]==embedding\_matrix[333])

[复制代码](javascript:void(0);)

**词向量矩阵维度：**

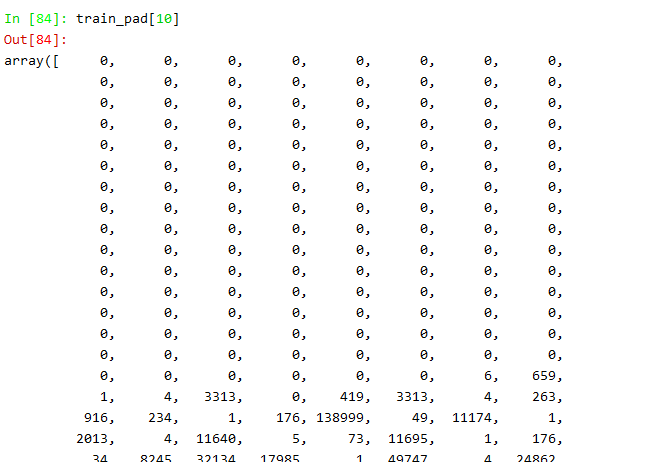
https://img2018.cnblogs.com/blog/1250855/201901/1250855-20190129110721358-1447736513.png

**4） padding（填充）和truncating（修剪）**

　　我们把问转换成token（索引）后，每一串索引的长度都不相等，所以为了方便模型的训练我们需要将索引的长度标准化，上面我们选择了使用236这个可以涵盖95%的训练样本的长度，接下来进行padding和truncating，我们一个采用‘pre’的方法，在文本索引的前面填充0。

# 返回一个numpy array

train\_pad = pad\_sequences(train\_tokens, maxlen=max\_tokens, padding="pre", truncating="pre")



**准备目标向量：**

# 准备target向量，前2000个位1，后2000个位0

train\_target = np.concatenate((np.ones(2000), np.zeros(2000)))

https://img2018.cnblogs.com/blog/1250855/201901/1250855-20190129112232896-1615953936.png

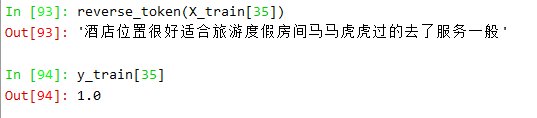
训练样本和测试样本分离，使用90%的样本来做训练，10%的样本用来做测试：

# 进行训练和测试样本的分割

from sklearn.model\_selection import train\_test\_split

# 90用作训练，正面和负面打乱

X\_train, X\_test, y\_train, y\_test = train\_test\_split(train\_pad, train\_target, test\_size=0.1, random\_state=12)



**5）使用Keras搭建神经网络模型（LSTM），模型的第一层是Embedding层**

**end**

# 综述论文：情感分析中的深度学习（内容抽象）

[**机器之心**](https://www.zhihu.com/org/ji-qi-zhi-xin-65)<https://zhuanlan.zhihu.com/p/33325977>

[**​**](https://www.zhihu.com/question/48509984)**作者：Lei Zhang、Shuai Wang、Bing Liu**

**机器之心编译**

近年来，深度学习有了突破性发展，NLP 领域里的情感分析任务逐渐引入了这种方法，并形成了很多业内最佳结果。本文中，来自领英与伊利诺伊大学芝加哥分校的研究人员对基于深度学习的情感分析研究进行了详细论述。

情感分析或观点挖掘是对人们对产品、服务、组织、个人、问题、事件、话题及其属性的观点、情感、情绪、评价和态度的计算研究。该领域的开始和快速发展与社交媒体的发展相一致，如评论、论坛、博客、微博、推特和社交网络，因为这是人类历史上第一次拥有如此海量的以数字形式记录的观点数据。早在 2000 年，情感分析就成为 NLP 中最活跃的研究领域之一。它在数据挖掘、Web 挖掘、文本挖掘和信息检索方面得到了广泛的研究。实际上，因其对商业和社会的整体重要性，它已经从计算机科学扩展到管理学和社会学，如营销、金融、政治学、传播学、健康科学，甚至历史学。这种发展原因在于观点是几乎所有人类活动的核心，是人类行为的重要影响因素。我们的信念、对现实的感知，以及我们所做的决策在很大程度上依赖于别人看到和评价世界的方式。因此，我们在做决策的时候，通常会寻求别人的意见。不只是个人，组织也是如此。

现有研究已经产生了可用于情感分析多项任务的大量技术，包括监督和无监督方法。在监督方法中，早期论文使用所有监督机器学习方法（如支持向量机、最大熵、朴素贝叶斯等）和特征组合。无监督方法包括使用情感词典、语法分析和句法模式的不同方法。现有多本综述书籍和论文，广泛地涵盖了早期的方法和应用。

大约十年前，深度学习成为强大的机器学习技术，在很多应用领域产生了当前最优的结果，包括计算机视觉、语音识别、NLP 等。近期将深度学习应用到情感分析也逐渐变得流行。本文首先概述深度学习，然后对基于深度学习的情感分析进行综述。

**论文：Deep Learning for Sentiment Analysis: A Survey**



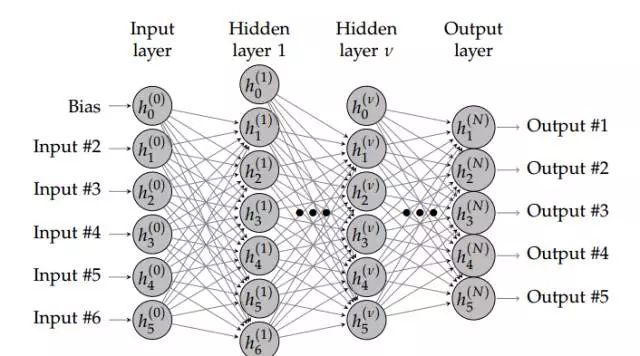
论文链接：[https://arxiv.org/pdf/1801.07883.pdf](https://link.zhihu.com/?target=https%3A//arxiv.org/pdf/1801.07883.pdf)

作为一项学习数据的多层特征或表征的强大机器学习技术，深度学习的出现实现了当前最优的预测结果。伴随着在诸多应用领域的成功，深度学习近年来也被广泛应用于情感分析。本论文首先概述深度学习，接着全面调研深度学习在情感分析领域的应用现状。

**情感分析中的基本模型**

**前馈神经网络**

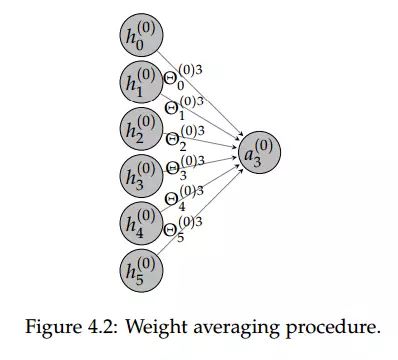
常规前馈神经网络（FNN）不考虑输入数据可能具备的任何特定结构。尽管如此，它仍是非常强大的机器学习工具，尤其是与先进的正则化技术一起使用时。这些正则化技术帮助解决人们处理「深度」网络时遇到的训练问题：神经网络有大量隐藏层，隐藏层非常难以训练（梯度消失和过拟合问题）。



*图 4.1：有 N + 1 层（N − 1 个隐藏层）的神经网络。浅层网络架构仅使用一个隐藏层。深度学习需要使用多个隐藏层，通常包含同样数量的隐藏神经元。数量大约是输入和输出变量数量的平均值。*

FNN 由一个输入层、一个（浅层网络）或多个（深层网络，因此叫作深度学习）隐藏层，和一个输出层构成。每个层（除输出层以外）与下一层连接。这种连接是 FNN 架构的关键，具有两个主要特征：加权平均值和激活函数。

加权平均过程，即将前一层给神经元的激励值和对应的权重矩阵相乘而得出后一个神经元的输入值，这一过程展示在下图 4.2 中，我们可以说前一层神经元的加权和就是后一层神经元的输入。



正式地，加权平均的过程可以使用如下方程式表达：



此外，每一层的隐藏神经元可以定义为：

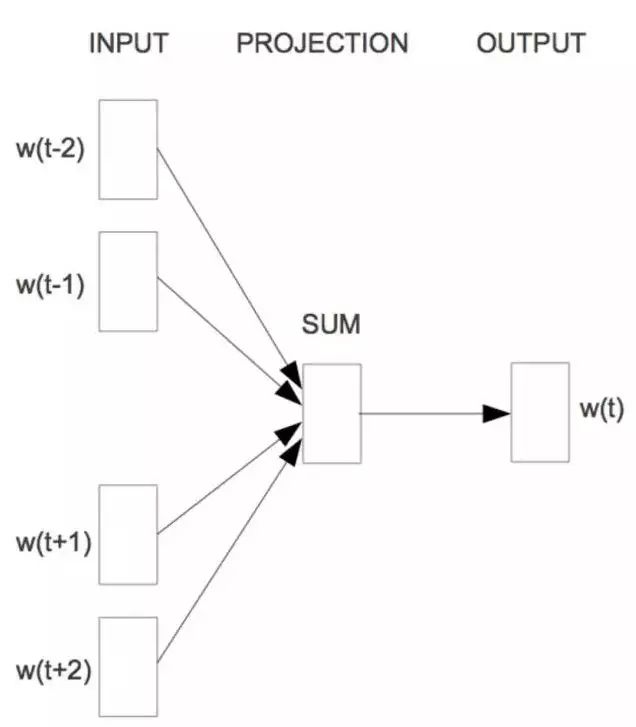
https://pic3.zhimg.com/80/v2-3af719796c0c9b3be4e2b4c208b8f7fe_hd.jpg

其中其中 v∈[0,N−1]、f∈[0,(F\_v+1)−1]、t∈[0,(T\_mb)− 1]。在这里 g 为非线性激活函数，是 FNN 另外一个十分重要的元素。因为激活函数的非线性属性，所以它允许预测任意的输出数据。

**Word2Vec 词嵌入**

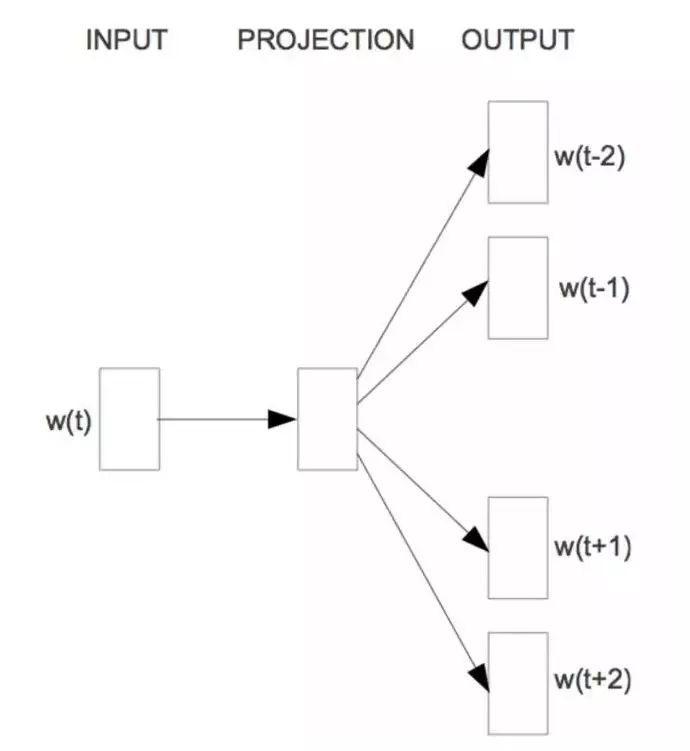
一般来说，Word2Vec 方法由两部分组成。首先是将高维 one-hot 形式表示的单词映射成低维向量。例如将 10，000 列的矩阵转换为 300 列的矩阵，这一过程被称为词**嵌入**。第二个目标是在保留单词上下文的同时，从一定程度上保留其意义。Word2Vec 实现这两个目标的方法有 skip-gram 和 CBOW 等，skip-gram 会输入一个词，然后尝试估计其它词出现在该词附近的概率。还有一种与此相反的被称为连续词袋模型（Continuous Bag Of Words，CBOW），它将一些上下文词语作为输入，并通过评估概率找出最适合（概率最大）该上下文的词。

对于连续词袋模型而言，Mikolov 等人运用目标词前面和后面的 n 个词来同时预测这个词。他们称这个模型为连续的词袋（CBOW），因为它用连续空间来表示词，而且这些词的先后顺序并不重要。



*连续的词袋（Mikolov 等人，2013 年）*

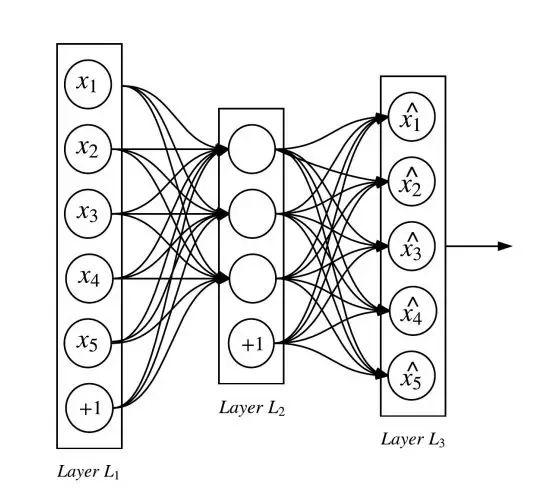
CBOW 可以看作一个具有先知的语言模型，而 skip-gram 模型则完全改变将语言模型的目标：它不像 CBOW 一样从周围的词预测中间的词；恰恰相反，它用中心语去预测周围的词：



*Skip-gram（Mikolov 等人，2013）*

**自编码器与降噪自编码器**

自编码器神经网络是一个三层神经网络，其目标是使输出值近似等价于输入值。下图展示了自编码器的一般架构：



*图 3：自编码器神经网络*

因为神经元使用了非线性激活函数，自编码器可以学习非线性表征。这令自编码器比主成分分析（PCA）或潜在语义分析（LSA）等线性方法要强大很多。

若我们将自编码器以层级的形式堆叠，那么高层的自编码器就使用低层自编码器的输出作为输入。这种堆叠的自编码器与受限玻尔兹曼机（RBM）是构建深度神经网络的早期方法。一旦我们以无监督的形式训练自编码器，那么描述 x（中间表征）多级表征的参数就能用来初始化监督式深度神经网络，这种神经网络在实验上已经证明要比随机初始化优秀。

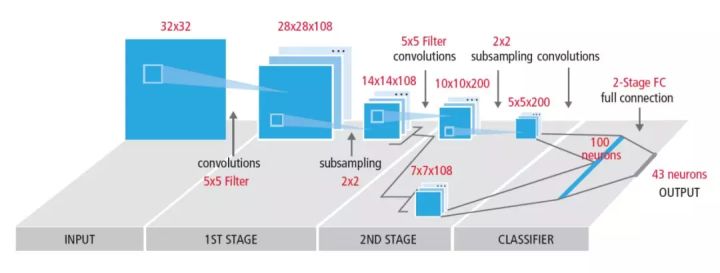
降噪自编码器（DAE）是自编码器的扩展，DAE 背后的思想是强制隐藏层发现更鲁棒的特征，并阻止自编码器简单地学习恒等变换。也就是说，模型应该在存在噪声时仍能重构输入。这种技术也体现在情感分析中，例如从文档中删除或添加一些文字不应该改变文档的语义。

**卷积神经网络**

CNN 非常擅长处理图像数据，如下图所示，它们一般由若干个卷积和池化操作组成，通常跟随着一个或多个全连接层（与传统的 FNN 层相似）。

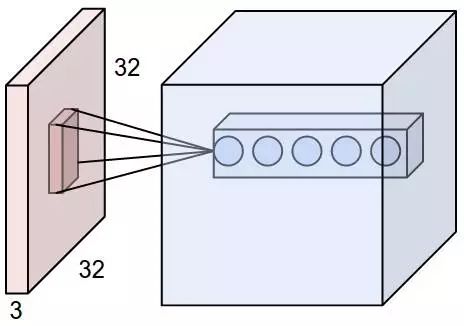
相比于全连接神经网络，卷积网络每一个单元都只会和上一层部分单元相连接。一般每个卷积层的单元都可以组织成一个三维张量，即矩阵沿第三个方向增加一维数据。例如 Cifar-10 数据集的输入层就可以组织成 32×32×3 的三维张量，其中 32×32 代表图片的尺寸或像素数量，而 3 代表 RGB 三色通道。

卷积神经网络中最重要的就是卷积层，卷积层试图将神经网络中的每一小块进行更加深入的分析，从而得出抽象程度更高的特征。一般来说通过卷积层处理的神经元结点矩阵会变得更深，即神经元的组织在第三个维度上会增加。



*图4：卷积神经网络*

为了理解卷积层，下图展示了卷积核或滤波器（filter）将当前层级上的一个子结点张量转化为下一层神经网络上的一个长和宽都为 1，深度不限的结点矩阵。下图输入是一个 32×32×3 的张量，中间的小长方体为卷积核，一般可以为 3×3 或 5×5 等，且因为要计算乘积，那么卷积核的第三个维度必须和其处理的图像深度（即输入张量第三个维度 3）相等。最右边的矩形体的深度为 5，即前面使用了五个卷积核执行卷积操作。这五个卷积核有不同的权重，但每一个卷积层使用一个卷积核的权重是一样的，所以下图五层特征中每一层特征都是通过一个卷积核得出来的，也就是该层共享了权重。

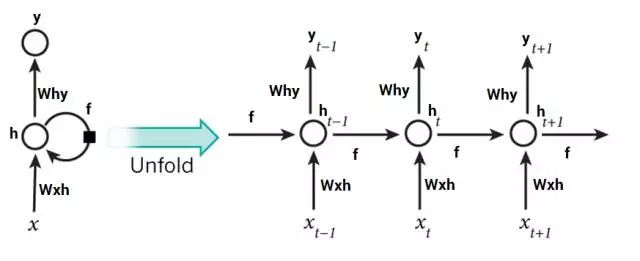


**循环神经网络**RNN

在传统的前馈神经网络中，所有的示例都被认为是独立的，它们没有时间关联性。这种时间关联性是由循环神经网络实现的。一个典型的 RNN 结构如下：



如果将其展开，它会变成这样：

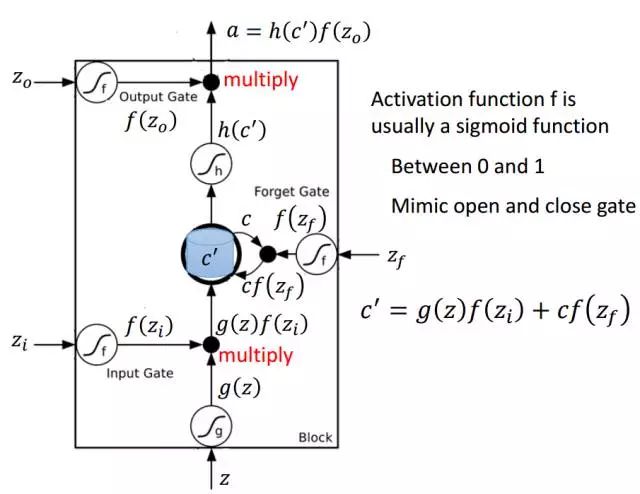


在这些图表中， x\_t 是时间序列上的输入，而 h\_t 是循环过程中的隐藏状态。我们看到 f 重复作用于不同时间步上的隐藏状态，并将它传入下一个时间步中，这就是 RNN特有的方式。基本上，你能输入句子中的词或者甚至是像 x\_t 这样的字符串中的字符，然后通过该循环神经网络它会得出一个 y\_t。  
目标是用 y\_t 作为输出，并将它与你的测试数据（通常是原始数据的一个小子集）比较。然后你会得出你的误差率。比较完之后，有了误差率，你就能使用一种叫随时间反向传播（BPTT）的技术。BPTT 返回检查这个网络，并基于误差率调整权重。这样也调整了这个网络，并让它学习去做得更好。

**LSTM 网络**

下面我们简要地向读者介绍 LSTM 单元选择记忆或遗忘的具体处理流程。

以下是 LSTM 单元的详细结构，其中 Z 为输入部分，Z\_i、Z\_o 和 Z\_f 分别为控制三个门的值，即它们会通过激活函数 f 对输入信息进行筛选。一般激活函数可以选择为 Sigmoid 函数，因为它的输出值为 0 到 1，即表示这三个门被打开的程度。



*图片来源于李弘毅机器学习讲义。*

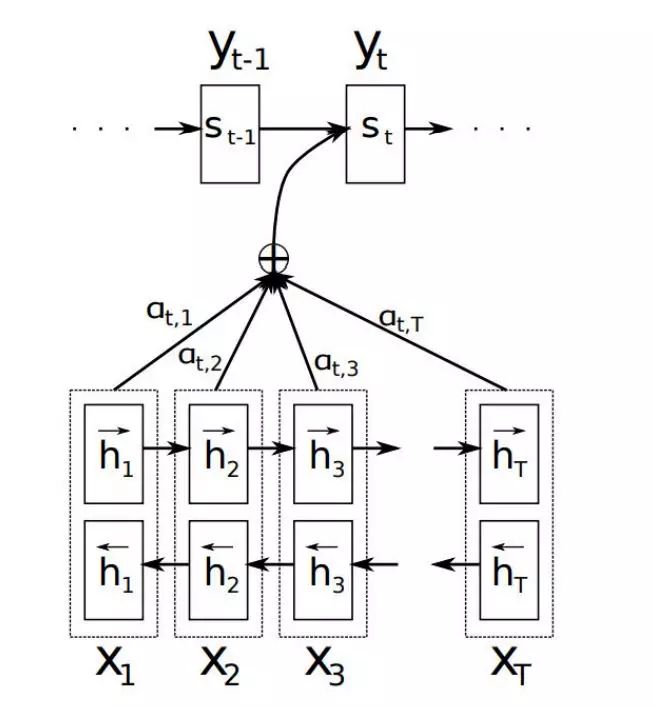
若我们输入 Z，那么该输入向量通过激活函数得到的 g(Z) 和输入门 f(Z\_i ) 的乘积 g(Z) f(Z\_i ) 就表示输入数据经筛选后所保留的信息。Z\_f 控制的遗忘门将控制以前记忆的信息到底需要保留多少，保留的记忆可以用方程 c\*f（z\_f）表示。以前保留的信息加上当前输入有意义的信息将会保留至下一个 LSTM 单元，即我们可以用 c' = g(Z)f(Z\_i) + cf(z\_f) 表示更新的记忆，更新的记忆 c' 也表示前面与当前所保留的全部有用信息。我们再取这一更新记忆的激活值 h(c') 作为可能的输出，一般可以选择 tanh 激活函数。最后剩下的就是由 Z\_o 所控制的输出门，它决定当前记忆所激活的输出到底哪些是有用的。因此最终 LSTM 的输出就可以表示为 a = h(c')f(Z\_o)。

**RNN 与注意力机制**

一般来说，我们可能会认为双向 RNN 与 LSTM 就能处理数据中的长期依赖性。但是在实践中，时序数据的长期依赖性问题仍然很难处理。因此，Bahdanau 等人提出了注意力机制。

神经网络中的注意力机制受到人类视觉中注意力的启发，即人类视觉注意力能够聚焦到图像的特定区域，并在这个区域有非常高的分辨率，而在其它区域有较低的分辨率。在自然语言处理中，注意力机制允许模型根据输入文本以及它到目前为止已经生成的隐藏状态来学习要注意什么，而不像标准 RNN 与 LSTM 那样将全部原文本编码成固定长度的向量。

下图 8 展示了在双向 RNN 中使用注意力机制的方法。其中每个解码器输出的序列 y\_t 取决于所有输入状态的加权组合，而不只是如标准情况那样选择最后一个隐藏状态。a\_t,T 定义了每个输入的隐藏状态应该加权多少以结合为输出向量。例如，a\_2,2 有较大的值，那么它就代表着在第二个时间步上，解码器更多注意原语句中的第二个隐藏状态。所有的权重 a\_t,T 加和为 1，因此能保证输出值的归一化。



*图 8：双向循环神经网络中的注意力机制*

**记忆网络**

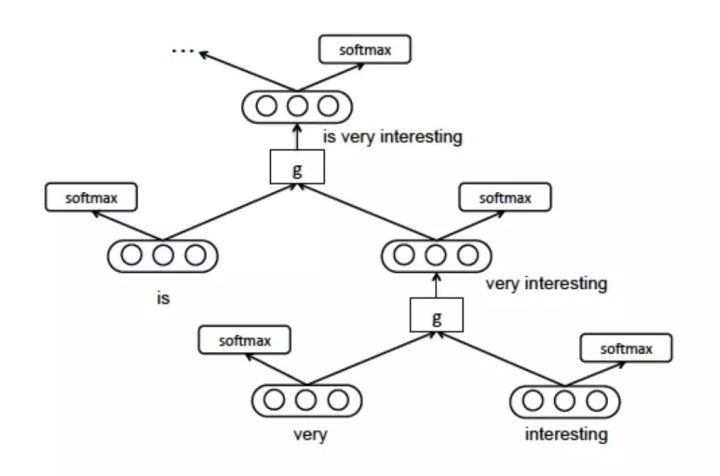
Weston 等人介绍了记忆网络（MemNN）这个概念，它能用于问答系统。记忆网络通过结合多个推断组件和长期记忆而执行任务，这些组件可以是多个神经网络，而长期记忆充当着动态知识库的角色。记忆网络基本的四个可学习或推断组件分别为：I 组件将输入数据转化为内部特征表示；G 组件在给定新的输入下更新旧的记忆；O 组件生成输出（同样是在特征表示空间中完成）；R 组件将输出特征转化为响应格式。例如，给定问答系统一系列语句和问题，MemNN 会从这些语句中抽取特征并生成答案。

在推断的过程中，I 组件一次只读取一条语句，并将它编码为向量表征。然后 G 组件基于当前的语句表征更新一小块记忆，在所有语句都处理完后，记忆网络就生成了一个记忆矩阵（每一行表示一个语句），该矩阵储存了从语句中抽取的语义。对于问题，记忆网络会将它编码为向量表征，然后 O 组件使用向量从记忆中选择一些相关的证据，并生成一个输出向量。最后，R 组件将输出向量作为输入，并输出最终响应。

**递归神经网络**

词嵌入是将单词表示成低维的稠密的实数向量。自从词向量技术的提出，到目前为止已经有很多方法来得到句法和语义方面的向量表示，这种技术在 NLP 领域发挥着重要的作用。

如何用稠密的向量表示短语，这是使用词向量的一个难题。在成分分析中，我们一般使用递归神经网络 (Recursive Neural Network) 来解决这个问题。递归神经网络是一种通用的模型，用来对句子进行建模。句子的语法树中的左右子节点通过一层线性神经网络结合起来，根节点的这层神经网络的参数就表示整句句子。递归神经网络能够给语法树中的所有叶子节点一个固定长度的向量表示，然后递归地给中间节点建立向量的表示。



*图 9：递归神经网络*

**情感分析任务**

我们现在开始概述情感分析中的深度学习应用。但在此之前，我们首先简单介绍主要的情感分析任务。若想了解更多细节，请参考 Liu 写的关于情感分析的书。

研究者主要在三个粒度级别上研究情感分析：文档级、语句级和 aspect level。文档级情感分类将观点鲜明的文档（例如，产品评论）分类为整体积极的或消极的观点。它将整个文档当做基本的信息单元，并假定文档是观点鲜明的，包含对单个实体（例如，某个型号的手机）的观点。语句级情感分类对文档内单独的语句进行分类。然而，单独的语句不能假定为观点鲜明的。

传统上，人们首先将一个语句分类为观点鲜明的（或相反），即主观性分类。然后观点鲜明的语句进一步被分类为积极的或消极的。语句级情感分类可以被形式化为三类分类问题，即判断某语句是中性的、积极的或消极的。和文档级、语句级情感分类相比，aspect level 情感分析或基于 aspect 的情感分析更加细粒化。它的任务是提取和总结人们对某实体的观点以及实体（也被称为目标）的特征。例如一篇产品评论，aspect level 情感分析的目的是分别总结对产品不同方面的积极和消极观点，虽然对产品的总体情感可能是倾向积极的或消极的。

基于 aspect 的情感分析由多个子任务构成，例如 aspect 提取、实体提取和 aspect 情感分类。例如，句子「the voice quality of iPhone is great, but its battery sucks」的实体提取应该识别「iPhone」作为实体，而 aspect 提取需要识别「voice quality」和「battery」作为两个 aspect。aspect level 情感分类需要将对音质的评论分类为积极的，将对电池续航的评论分类为消极的。出于简洁性，大多数算法将 aspect 提取和实体提取结合起来，称为 aspect 提取或情感/观点目标提取。

除了这些核心任务以外，情感分析还研究了情绪分析、嘲讽检测、多语言情感分析等。在接下来的章节中，我们将概述所有这些情感分析任务中的深度学习应用。

**文档级情感分类**

文档级情感分类是指为观点型文档标记整体的情感倾向／极性，即确定文档整体上传达的是积极的还是消极的观点。因此，这是一个二元分类任务，也可以形式化为回归任务，例如为文档按 1 到 5 星评级。一些研究者也将其看成一个五类分类任务。

情感分类通常被当做文档分类的特殊案例。在这种分类任务中，文档表征是很重要的部分，需要反映出文档字里行间所传达的原始信息。传统上，词袋模型（BoW）通过将文档看成其中单词的袋装形式，被用于在 NLP 和文本挖掘中生成文本表征。通过 BoW，文档被转换成固定长度的数值特征向量，其中每个元素可能代表词的存在（没出现或出现）、词频或 TF-IDF 分数。向量的维度等于词汇量大小。用 BoW 表征的文档向量通常是很稀疏的，因为单个文档仅包含少量的词汇。早期的神经网络使用的基本是这种特征设定。

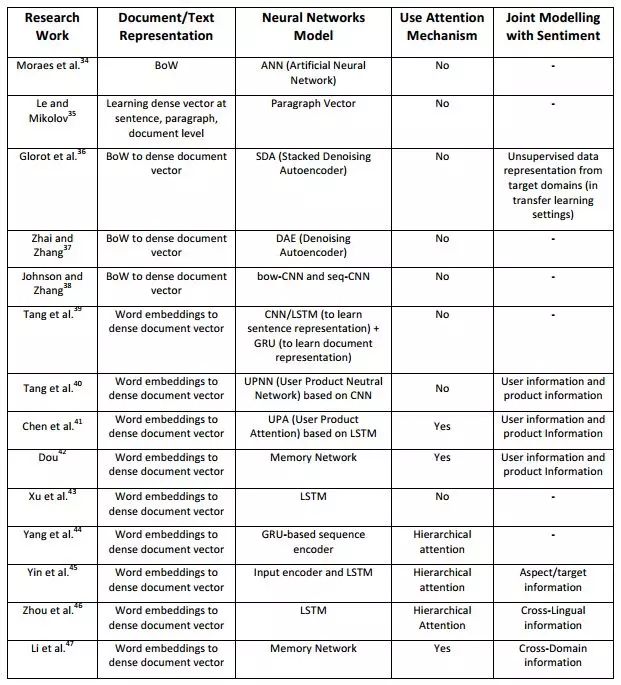
虽然 BoW 很常用，它也有一些缺点。首先，BoW 模型忽略词的顺序，这意味着包含相同单词的两个文档的表征是完全相同的。BoW 的扩展版本 Bag-of-N-Grams 在短文本（n-gram）中考虑词序，但仍然存在数据稀疏性和高维度的缺陷。其次，BoW 几乎不能编码词的语义。例如，在 BoW 中，单词「smart」、「clever」和「book」之间的距离是相同的，但在语义上，相比「book」，「smart」应该更接近于「clever」。

为了克服 BoW 的缺陷，人们提出了基于神经网络的词嵌入技术以生成密集向量（或低维向量）用于词表征，从而在某种程度上可以编码单词的某些语义和句法属性。以词嵌入作为词的输入，可以利用神经网络得到文档的密集向量（或称为密集文档向量）表征。

除了以上两种方法，实际上也可以直接用 BoW 学习密集文档向量。我们在表 2 中区分了相关研究使用的不同方法。

当文档被适当地表征时，我们可以通过传统的监督学习方式，用多种神经网络模型进行情感分类。在某些案例中，神经网络可能只被用于提取文本特征或文本表征，然后这些特征被馈送到其它非神经网络的分类器（如 SVM），以获得最终的全局最优分类器。神经网络和 SVM 的特性以某种方式彼此互补，从而能结合各自的优势。

除了复杂的文档/文本表征之外，研究者还利用数据特征（如产品评论）进行情感分类。一些研究者发现产品评论对情感和其它附加信息（例如，用户信息和产品信息）进行联合分类建模很有帮助。此外，由于文档通常包含长期依赖关系，注意力机制也经常用于文档级情感分类。我们在表 2 中总结了已有的技术。



*表 2：文档级情感分类的深度学习方法。*

**语句级的情感分类**

语句级情感分类用来标定单句中的表达情感。正如之前所讨论的，句子的情感可以用主观性分类和极性分类来推断，前者将句子分为主观或客观的，而后者则判定主观句子表示消极或积极的情感。在现有的深度学习模型中，句子情感分类通常会形成一个联合的三类别分类问题，即预测句子为积极、中立或消极。

与文档级的情感分类相同，神经网络生成的语句表征对于语句级的情感分类也非常重要。另外由于句子相对文档而言较短，因此可以使用一些语法和语义信息（如解析树、观念词典和词性标签）来帮助分类。其他一些信息如评测打分、社会关系和跨域信息也可以考虑在内。例如，社会关系已被用于探索社交媒体数据中（如推文）的情感。

在早期的研究中，解析树（提供了一些语义和语法信息）与原始词一同用作神经模型的输入，这意味着我们可以更好地推断情感构成。但在那之后，CNN 和 RNN 成为主流，它们不需要利用解析树从句子中提取特征。取而代之的是，CNN 与 RNN 使用词嵌入（已经编码了一些语义和语法信息）作为输入。此外，CNN 和 RNN 模型架构也可以帮助我们学习语句内词间的固有联系。

**Aspect Level 情感分类**

与文档级和语句级的情感分类不同，aspect level 情感分类同时考虑了情感信息和目标信息（情感一般都会有一个目标）。如前所述，目标通常是一个实体或实体特征。出于简洁性，实体和实体特征通常都称为特征（aspect）。给定一个句子和目标特征，aspect level 情感分类可以推断出句子在目标特征的情感极性／倾向。例如，句子「the screen is very clear but the battery life is too short.」中，如果目标特征是「screen」，则情感是积极的，如果目标特征是「battery life」，则情感是消极的。下一节将讨论自动特征提取或目标提取。

Aspect Level 情感分类很有难度，因为建模目标与上下文的语境词的语义相关性很难。不同的语境词对句子在目标特征的情感极性有不同的影响。因此，使用神经网络构建学习模型时，捕捉目标词和语境词之间的语义关系非常必要。

使用神经网络的 aspect level 情感分类有三个重要任务。第一个任务是表示目标的语境词。该问题可以使用前两节提到的文本表示方法来解决。第二个任务是生成目标表示，其可与语境词进行恰当地互动。通常的解决方案是学习目标嵌入（与词嵌入类似）。第三个任务是识别特定目标的重要情感语境词。例如，在句子「the screen of iPhone is clear but batter life is short」中，「clear」是「screen」的重要语境词，「short」是「battery life」的重要语境词。近期该任务通过注意力机制得到解决。尽管很多深度学习技术可用于处理 aspect level 情感分类，但文献中仍然没有主导性技术。

**带有词嵌入的情感分析**

很明显词嵌入在深度学习情感分析模型中扮演了重要角色。研究也表明，即使不使用深度学习模型，词嵌入也可以在不同任务中用作非神经网络学习模型的特征。因此，该部分特别强调了词嵌入对情感分析的贡献。

我们首先介绍了情感编码词嵌入的工作。对于情感分析，直接使用 CBOW 或 Skip-gram 等常规的单词方法学习语境中的词嵌入可能会遇到问题，因为具有相似语境但情感极性相反（例如，「好」或「坏」）的单词可能被映射到嵌入空间的相近向量。因此，人们提出了情感编码词嵌入方法。Mass el al.101 学习了可以捕捉语义和情感信息的词嵌入。Bespalov et al.102 表明，n-gram 模型结合潜在表征将为情感分类提供更合适的嵌入。通过把语句的情感监督作为正则化项，Labutov and Lipson103 将带有 logistic 回归的现有词嵌入进行重嵌入。

**用于情感分析的多模态数据**

多模态数据已被用于情感分析，因为其比文本提供了更多的信息。深度学习模型把输入映射到一些特征空间，来自多模态数据的不同形式的输入也可以被这些模型投射到一些联合潜在空间或表征。因此，使用深度学习模型处理多模态数据的趋势不断增长。

例如 Wang et al. 提出一个 CNN 结构的深度网络，命名为深度耦合形容词与名词神经网络（DCAN），可用于视觉情感分类。DCAN 的核心思想是利用形容词和名词性文本描述，把它们看作两个（弱）监督信号以学习两个中间情感表征，然后结合学习的表征并用于情感分类。

***本文为机器之心编译，转载请联系本公众号获得授权。***编辑于 2018-01-26